ROC Analysis of IR Segmentation Techniques

THESIS

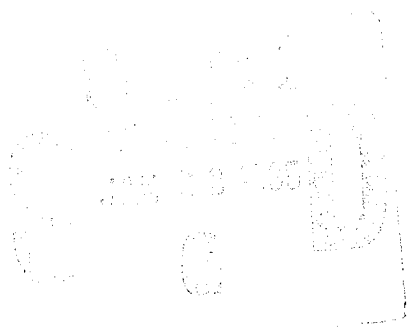
Georgia Kay Harrup
Second Lieutenant, USAF

AFIT/GE/ENG/94D-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/94D-15



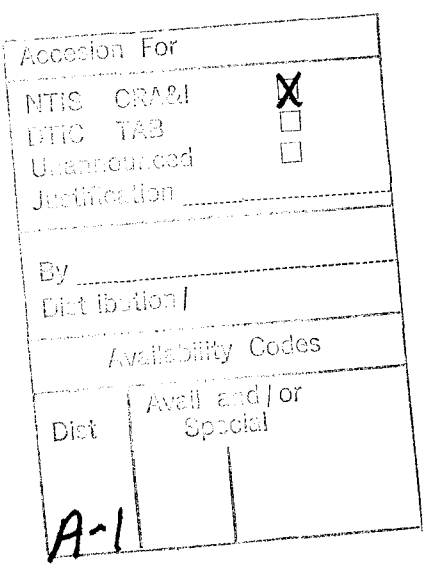
ROC Analysis of IR Segmentation Techniques

THESIS

Georgia Kay Harrup
Second Lieutenant, USAF

AFIT/GE/ENG/94D-15

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.



AFIT/GE/ENG/94D-15

ROC Analysis of IR Segmentation Techniques

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Georgia Kay Harrup, BSEE
Second Lieutenant, USAF

Dec, 1994

Acknowledgements

First, I would like to thank my advisor, Dr. Steven Rogers, along with the rest of my committee, Dr. Mark Oxley and Capt. Dennis Ruck, for their involvement in this research. I'd also like to thank Lt. Curtis Martin for explaining his methods to me and giving me a starting point for this research. Next I'd like to give a big thanks to Dan Zambon and Dave Doak for all their work in keeping the computers going. Thanks to Dave for helping me through all the problems I had with Khoros.

Next, I'd like to thank all my friends and colleagues who helped me make it through the last year and a half. Thanks to everyone who helped me at work. Also, thanks to everyone who dragged me out of my apartment every once in a while and made me have some fun.

Finally, I'd like to thank my husband Mark for being as supportive as he could from a thousand miles away.

Georgia Kay Harrup

Table of Contents

List of Figures

List of Tables

Abstract

Receiver Operating Characteristic (ROC) curves are used to compare the effectiveness of IR image processing techniques. Two non-parametric error estimation techniques (k-Nearest Neighbor and Parzen Window) are used to create estimates of the probability density functions for the data. These pdfs are used in the creation of the ROC curves for both resubstitution and leave-one-out estimates. These estimates generate the upper and lower bounds, respectively, on the ROC curves. The ROC curve analysis is performed on the outputs of various image processing techniques and the resulting ROC curves are used to compare the techniques. Of the image processing techniques used in this thesis, the close minus open (CMO) morphological filter operation produced the best results.

ROC Analysis of IR Segmentation Techniques

I. Introduction

1.1 Background

Image processing, segmentation, and target recognition are important areas of research to the Air Force. The object of image processing, segmentation, and target recognition is to automatically identify regions of the image that contain targets, without requiring human intervention. This will allow the pilot of an aircraft to concentrate on other things such as maneuvering through enemy fire. Automatic segmentation will also identify targets that may be difficult for the pilot to see because they are either too far away or camoflauged. Typical infrared images are shown in Figures 1.1 and 1.2.

There are many image processing techniques available, such as spatial feature extraction, morphological filters, and hit-miss transforms, but none is perfect in every situation. These techniques are based upon probabilities and statistics and, therefore, have errors associated with them. Because there are errors, a method for comparing the efficiency of each technique must be found. One way to do this is through Receiver Operating Characteristic (ROC) curves.

ROC analysis was based on statistical decision theory and developed in signal detection theory [13, 19]. In the past, the performance of classification systems was measured by the percentage of correct decisions, but that "percent correct" does not account for the false-positive and false-negative errors involved [13]. For example, if 5% of people have a particular disease, then a system can be 95% accurate by calling everyone negative. The problem with this number is that it does not take into account that the system fails to find any positives. ROC analysis solves this problem

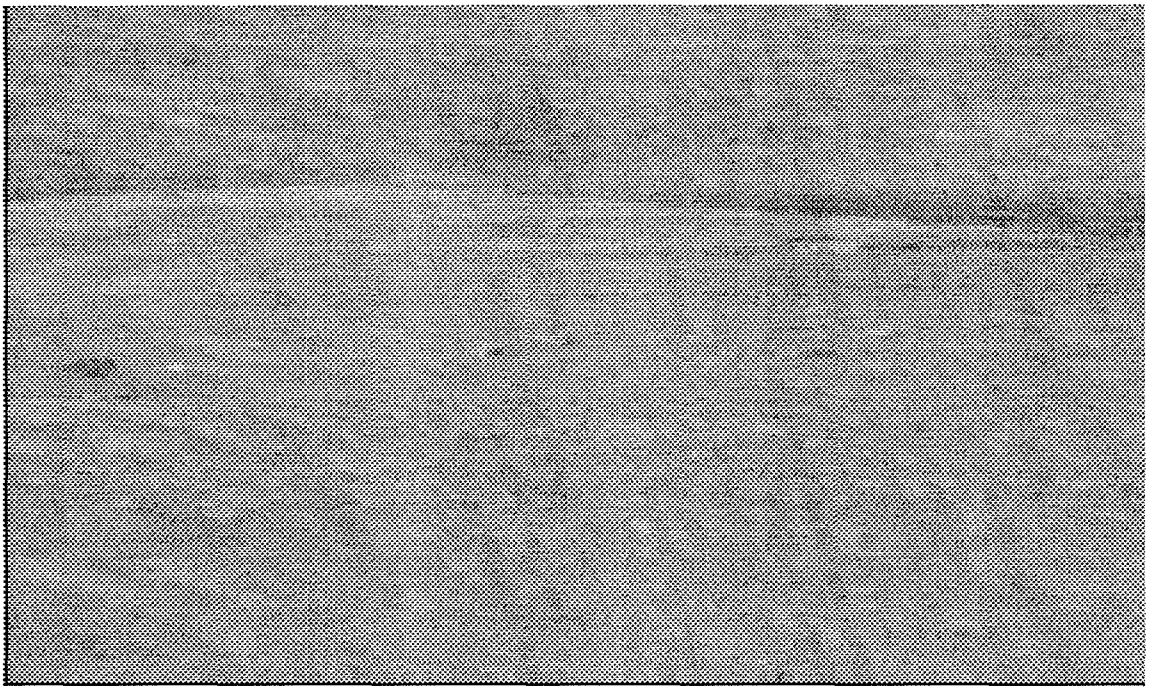


Figure 1.1 Typical Infrared Image (range=5775 ft).

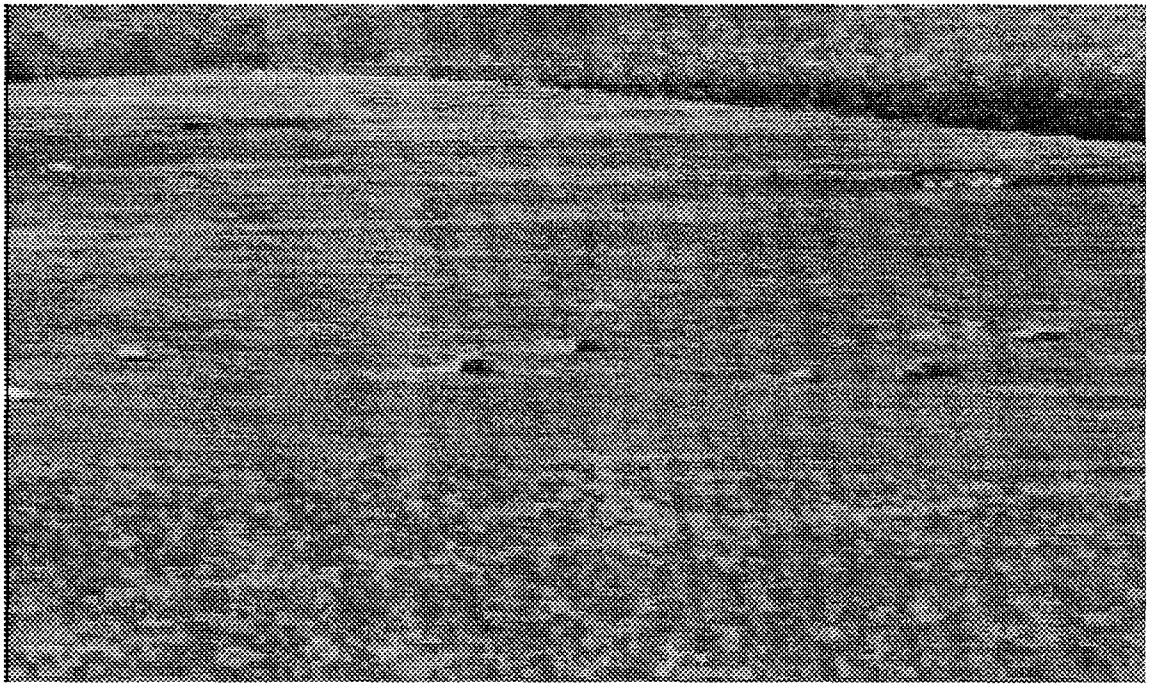


Figure 1.2 Typical Infrared Image (range=4100 ft).

by plotting true positive fractions (TPF) against false positive fractions (FPF) [13]. In the area of target recognition and in this thesis, these are the probabilities of hit and false alarm, respectively.

In order to create the ROC curves from finite sample sets, the probability density functions need to be estimated. Two common non-parametric density estimation techniques are the k-Nearest Neighbor estimation proposed by Fix and Hodges, and the Parzen Window approach proposed by Parzen [6, 15]. These non-parametric techniques for estimating the pdfs are used because they do not assume a functional form for the pdf.

Resubstitution and leave-one-out methods are used to create the upper and lower bounds, respectively, on the ROC curves. The resubstitution method involves using all available samples to train the classifier, and then testing with those same samples. Lachenbruch proposed the leave-one-out method which involves using all but one sample to train the classifier, testing with that one sample, and repeating the process for all data samples [10]. Fukunaga and Hummels showed that the resubstitution and leave-one-out techniques can be applied to k-NN and Parzen classifiers to obtain upper and lower bounds on the Bayes error rate [7]. Their technique is used in this research to find the density estimations from which the ROC curves are created.

1.2 Problem Statement

This thesis focuses on estimating the ROC curves using various segmentation techniques to segment infrared images, and then processing the segmented images to classify the pixels into two classes.

1.3 Scope

Actual infrared imagery taken by aircraft flying over targets in a desert background is used in producing the ROC curves. This imagery was obtained from

the Wright Laboratories Automatic Target Recognition Branch at Wright-Patterson AFB. A completely different set of imagery was obtained for the University of Cincinnati. This imagery contained x-ray images of breast cancer tumors, both malignant and benign. Although the ROC curves generated in this thesis apply specifically to the images used, the software can be used to obtain ROC curves for a variety of targets in various backgrounds. This thesis concentrates on using only k-NN and Parzen Window techniques to produce probability estimates for a two-class problem: target and non-target.

1.4 Approach

The approach taken consists of first implementing current image processing techniques: morphology and spatial feature extraction. Then Parzen Window and k-Nearest Neighbor techniques, described by Fukunaga and Hummels and validated by Martin, are used to estimate probability density functions from the images [7, 11]. From the probabilities obtained, ROC curves are constructed.

1.5 Research Objectives

The research objectives for this thesis are as follows:

1. Implement image processing techniques: morphology and spatial feature extraction [3, 8].
2. Implement Parzen and k-NN error estimation approaches to construct ROC curves for each processing technique [7, 11, 12].
3. Show that the ROC curve analysis is consistent from one image to the next.
4. Develop an understanding of how each image processing and error estimation technique works, as well as how to interpret the results.

1.6 Thesis Organization

The rest of this thesis is organized as follows: Chapter II contains the theory behind the probability and error estimations, and each image processing technique. Chapter III discusses the methods used to implement the processing techniques and error estimations. Chapter IV presents the results obtained from applying each technique. Finally, Chapter V gives an overall summary of this research.

1.7 Summary

Image segmentation and target recognition are ongoing areas of research. This thesis investigates image processing techniques as well as a method for measuring their performances in segmentation applications.

II. Theory

This chapter describes the theory behind the methods used in this thesis. The following topics are discussed:

- Probability theory
- Non-parametric density estimation
- Receiver Operator Characteristic curves
- Morphology.

2.1 Probability Theory

Probability theory is a mathematical method used to describe random phenomena which are not deterministic, but can be approximately described by the relative frequency of the occurence of the possible outcomes [18]. A simple example of this is the tossing of a fair coin. Although the outcome on any given toss cannot be predicted, it is likely that after many tosses, approximately half of the outcomes will be heads and the other half tails.

The probability density function (pdf) is a representation of the distribution of the outcomes of a random experiment. It shows the probability for observing any outcome on a given trial of the experiment. The pdf of a single real variable has the following properties:

$$f(x) \geq 0 \quad \text{for all } x \tag{2.1}$$

$$\int_{-\infty}^{\infty} f(x)dx = 1. \tag{2.2}$$

The pdf for the coin toss mentioned above can be seen in Figure 2.1. Another example, this time a continuous pdf, can be seen in Figure 2.2. In this case, the probability of the outcome of the experiment falling between points a and b is the

shaded area described by

$$\int_a^b f(x)dx. \tag{2.3}$$

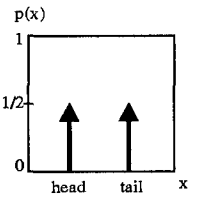


Figure 2.1 The probability density function for the coin toss.

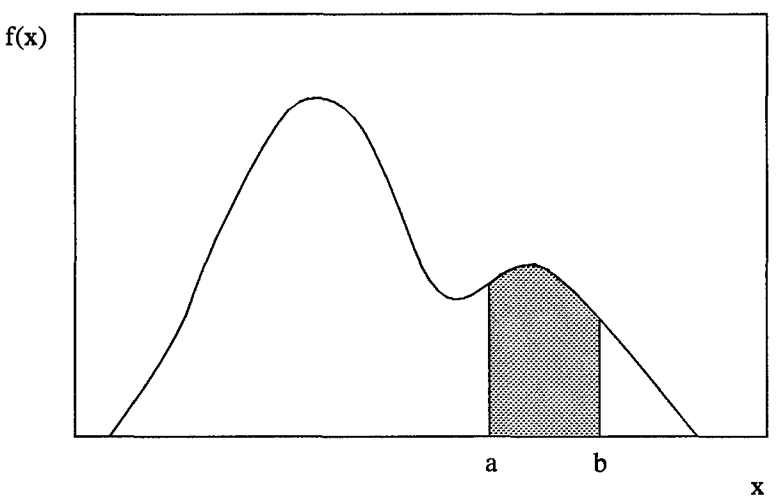


Figure 2.2 A continuous probability density function.

Once pdfs have been estimated, error probabilities can be found. When two conditional pdfs are plotted on the same axis as in Figure 2.3, a decision threshold (t) can be placed at any point on the axis and a decision rule can be declared. The decision rule used here is the Bayes decision rule where

$$P(s_1|x) \underset{S_2}{\overset{S_1}{\gtrless}} P(s_2|x) \tag{2.4}$$

or

$$\frac{p(x|s_1)P(s_1)}{p(x)} \underset{S_2}{\overset{S_1}{\gtrless}} \frac{p(x|s_2)P(s_2)}{p(x)} \tag{2.5}$$

[16]. For example, given the pdfs in Figure 2.3, the decision rule would be to choose S_1 if the outcome falls to the left of the decision threshold and S_2 if it falls to the right. In following this decision rule, the probability of error for S_1 (an S_1 point that is declared S_2) would be the area under the S_1 pdf that lies to the right of the threshold. Similarly, the probability of error for S_2 would be the area under the S_2 pdf that lies to the left of the threshold.

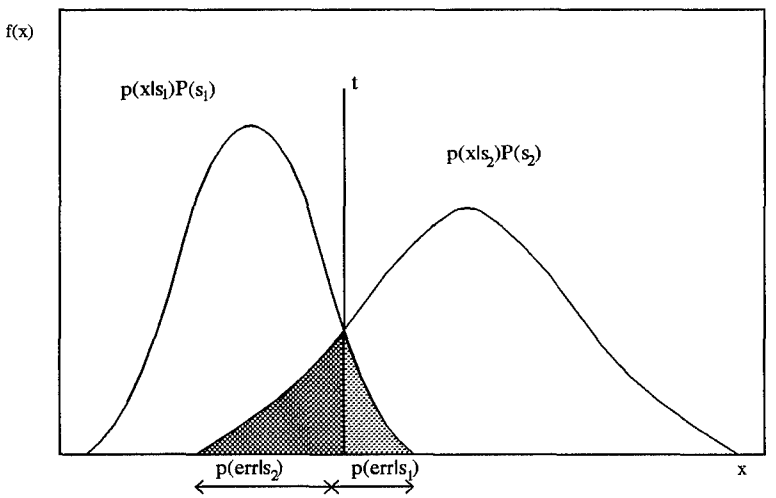


Figure 2.3 Example pdfs showing error probabilities.

2.2 Non-Parametric Density Estimation

Non-parametric density estimation is a way of estimating the pdf of a random variable given a finite number of samples without assuming a functional form. In this section, the two estimation techniques used in this thesis will be described: k-nearest neighbor and Parzen window.

2.2.1 k-Nearest Neighbor. The k-nearest neighbor (k-NN) technique for estimating pdfs is based on the volume created by enclosing the k nearest samples to

a data point by a contour of constant distance. If this volume is small, that means the points are closely spaced and the probability of an outcome falling in that region is high. Therefore, the pdf is large at that point. Conversely, if the volume is large, the points are spread out and there is a lower probability of an outcome falling in that region. Therefore, the pdf is small at that point. An example of this can be seen in Figure 2.4. (In actuality, the contour enclosing points along a one-dimensional line is only a length, but for illustration purposes a circle is used.)

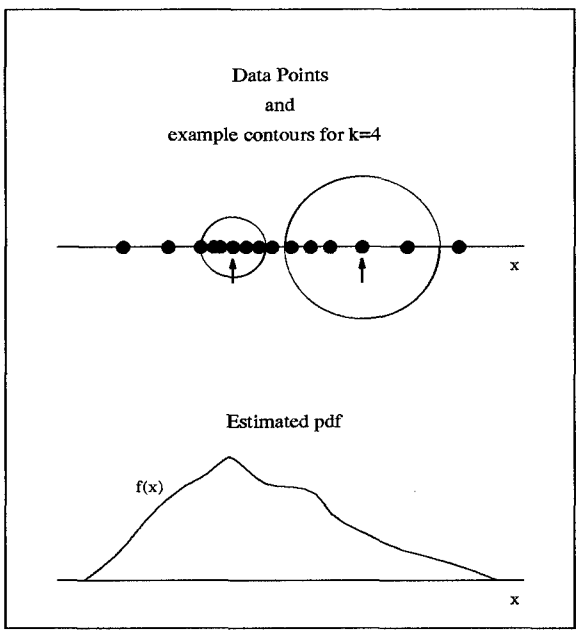


Figure 2.4 k-NN density estimation. This estimation for k=4 illustrates how the contours enclose the 4 nearest points to each chosen point.

The k-NN density estimation for class s_i at point x is

$$\hat{p}_i(x) = \frac{k-1}{N_i \; V_k^{(i)}(x)} \tag{2.6}$$

where N_i is the total number of samples, k is the number of neighbors, and $V_k^{(i)}(x)$ is the volume inside the surface of constant distance which encloses those k nearest neighbors [7, 11].

To calculate the volume inside the surface of constant distance, a distance metric must be chosen. The one used in this thesis is the squared Mahalanobis distance described by

$$d^2(x,y) = (x-y)^T \sum^{-1} (x-y) \tag{2.7}$$

where $\sum^{-1}$ is the inverse covariance matrix [11]. In n-dimensional space, a surface of constant distance is a hyper-ellipsoid having a volume

$$V_d \left|\sum\right|^{1/2} r_{ik}(x) \tag{2.8}$$

where

$$r_{ik}(x) = \sqrt{d_i^2(x, x_{k-NN}^{(i)})} \tag{2.9}$$

$$V_d = \begin{cases} \frac{\pi^{n/2}}{(n/2)!} & \text{n even} \\ \frac{2^n \pi^{(n-1)/2} \left(\frac{n-1}{2}\right)!}{n!} & \text{n odd} \end{cases} \tag{2.10}$$

[4, 11]. Substituting into Equation 2.6, the k-NN estimate of the class s_i pdf at point x is

$$\hat{p}_i(x) = \frac{k-1}{N_i V_d \left|\sum\right|^{1/2} \sqrt{[d_i^2(x, x_{k-NN}^{(i)})]^n}} \tag{2.11}$$

[11].

2.2.2 Parzen Window. In simple terms, the Parzen window technique for estimating pdfs is obtained by placing a small window function at every data point. When the areas of all the window functions are added together, the resulting function will approximately estimate the density function. A simple example is shown in Figure 2.5.

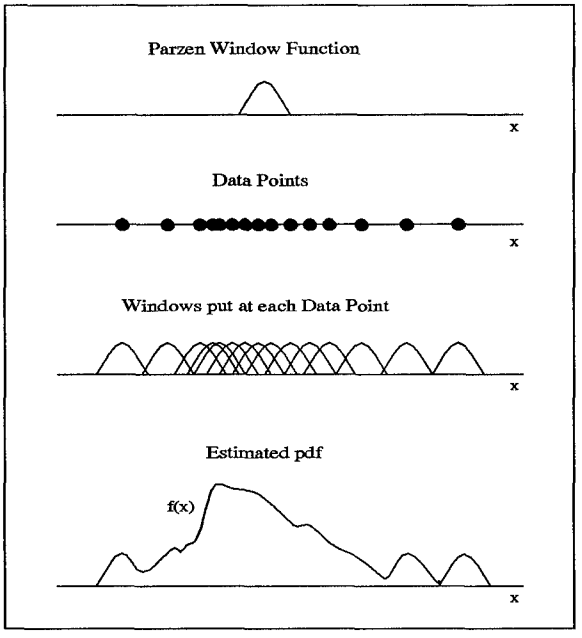


Figure 2.5 Parzen window density estimation.

Mathematically, the Parzen estimate of the conditional pdf at point x for class s_i containing data in n-dimensional space is

$$\hat{p}_i(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{1}{h^n} k\left(\frac{x - x_j^{(i)}}{h}\right) \tag{2.12}$$

where $k(\cdot)$ is a kernel (or window) function, h is a parameter that controls the spread of the window, N_i is the number of samples, and x_j are the samples [7, 11].

The window used in this thesis is a Gaussian with the following functional form:

$$f(x) = \frac{1}{(2\pi)^{n/2} \left|\sum\right|^{1/2}} \exp[-\frac{1}{2}(x - \mu)^T \sum^{-1} (x - \mu)] \tag{2.13}$$

where μ is a mean vector and $\sum$ is a covariance matrix [4, 11]. The equation for the window kernel is

$$k_i\left(\frac{x - x_j^{(i)}}{h}\right) = \frac{1}{(2\pi)^{n/2} \left|\sum_i\right|^{1/2} h} \exp\left[-\frac{(x - x_j^{(i)})^T \sum_i^{-1} (x - x_{(i)}^j)}{2h^2}\right] \tag{2.14}$$

where the h^2 varies the spread of the window [11]. Substituting into Equation 2.12, the Parzen density estimate with the Gaussian window function is

$$\hat{p}_i(x) = \frac{1}{N_i} \sum_{j=1}^{N-i} \frac{1}{h^n} \left[\frac{1}{(2\pi)^{n/2} |\sum_i|^{1/2} \; h} \exp\left(-\frac{d_i^2(x, x_j^{(i)})}{2h^2} \right) \right]. \qquad (2.15)$$

2.3 Receiver Operating Characteristic Curves

For an in depth description of ROC curves, see the works by Egan or Metz [5, 13]. In simple terms for this thesis, a ROC curve is the probability of a hit plotted versus the probability of false alarm. Given two overlapping pdfs, each point in the shared region will have both a probability of a hit and false alarm as demonstrated in Figure 2.6. For each point, the pair of probabilities are plotted together as in Figure 2.7 to obtain the ROC curve. Creating these curves for each image processing technique provides a means to measure the accuracy of each technique and make comparisons between different techniques.

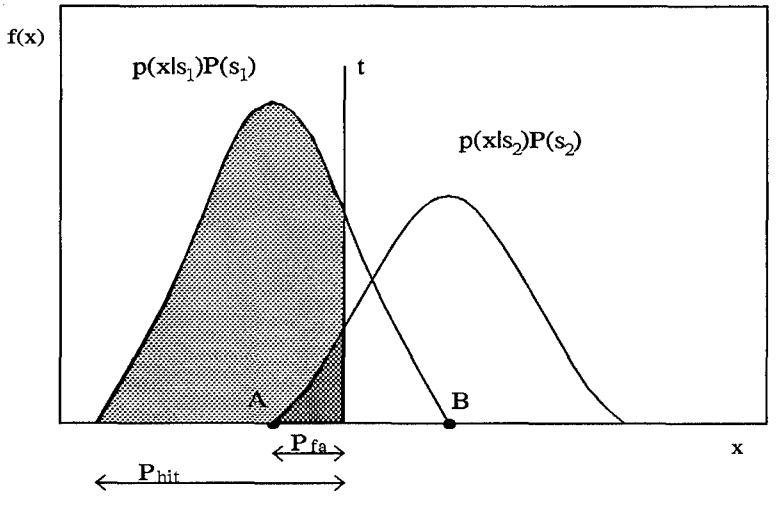


Figure 2.6 Probability of hit and false alarm for S_1 given two overlapping conditional pdfs.

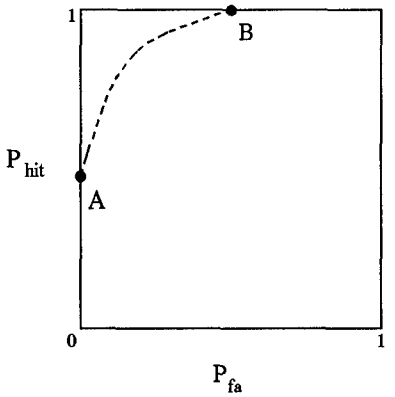


Figure 2.7 An example ROC curve. Shows the resulting ROC curve of the pdfs in Figure 2.6.

2.4 Morphology

Mathematical morphology is an image processing tool used to extract image components that are used to describe shapes in the image [8]. Set theory is used to describe mathematical morphology, and sets are used to represent the shapes of objects in the image. The following sections describe set theory and the basic functions of morphology.

2.4.1 Basic Definitions of Set Theory. In order to understand morphology, one needs to know some basic definitions dealing with set theory.

First of all, a set is a collection of objects called elements [14]. For this thesis, these elements are pixels in an image, so a set is a collection of pixels.

A subset is another set whose elements are also elements of a larger set [14]. This is denoted by $Y \subset Z$ if Y is a subset of Z.

The union of two sets is a new set whose elements are all the elements of the original two sets. For example, if set $Y = \{1, 2, 3, 4, 5\}$ and set $Z = \{2, 4, 6, 8\}$ then the union is $Y \cup Z = \{1, 2, 3, 4, 5, 6, 8\}$.

The intersection of two sets is a new set consisting of all elements that are common to both sets. Using sets Y and Z from above, the intersection is $Y \cap Z = \{2, 4\}$.

The following definitions were taken from the work by Gonzalez and Woods [8]. The complement of set A consists of all elements not in A and is defined as

$$A^c = \{x | x \notin A\}. \tag{2.16}$$

An example is shown in Figure 2.8.

The translation (shift) of set B by $x = (x_1, x_2)$ is defined as

$$(B)_x = \{c | c = b + x, \text{ for } b \in B\}. \tag{2.17}$$

The reflection of B is defined as

$$\hat{B} = \{x | x = -b, \text{ for } b \in B\}. \tag{2.18}$$

The difference of the two sets A and B is defined as

$$A - B = \{x | x \in A, \ x \notin B\} = A \cap B^c \tag{2.19}$$

and is also shown in Figure 2.8.

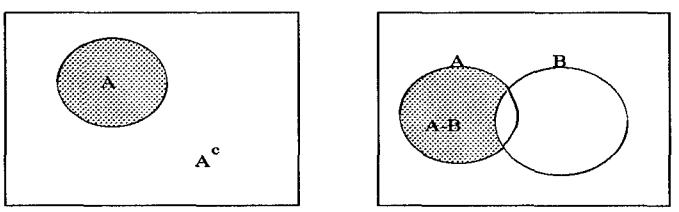


Figure 2.8 Venn diagrams showing the complement and difference.

For the morphology examples that follow, A is the image and B is the structuring element (kernel). For binary morphology, A and B are sets in the 2-D integer space with components $a = (a_1, a_2)$ and $b = (b_1, b_2)$.

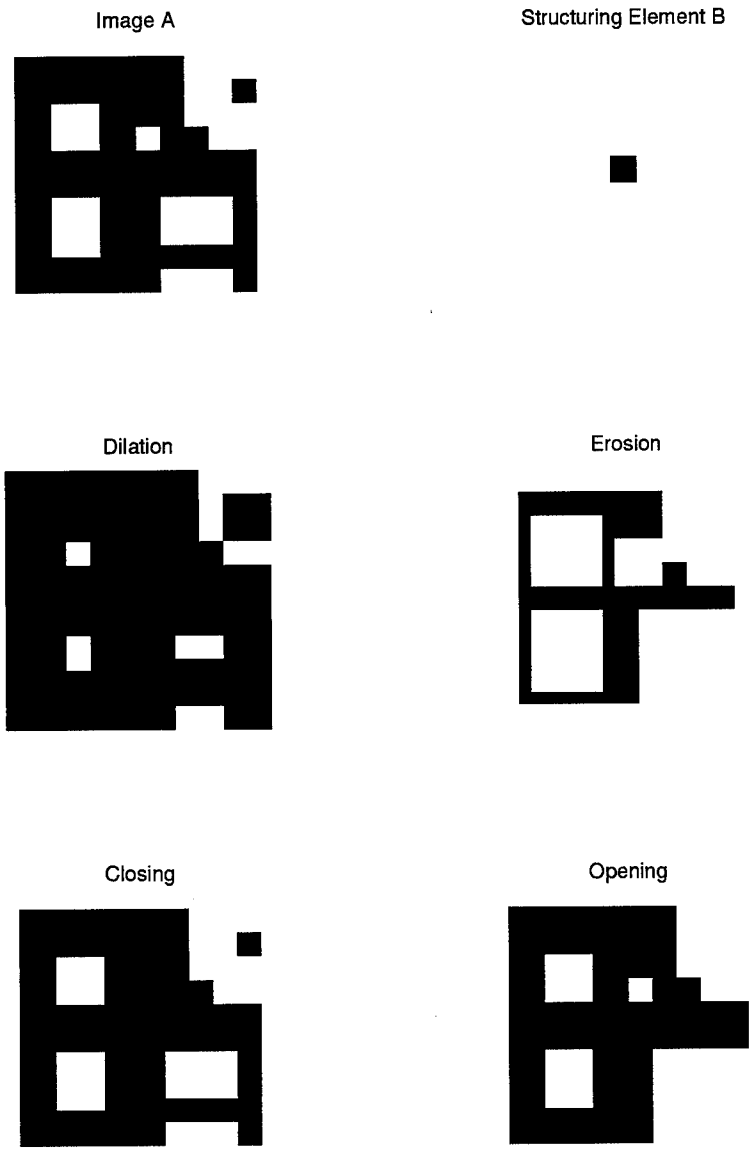


Figure 2.9 Example of binary morphology. (photo-negative: black=1, white=0)

2.4.2 Dilation. Dilation is a process that expands the image A by the structuring element B. The process consists of reflecting B and then finding the set of all shifts x where A and $\hat{B}$ overlap by at least one nonzero element. This dilation process is defined by the following equations:

$$A \oplus B = \{x|(\hat{B})_x \cap A \neq \emptyset\} \tag{2.20}$$

$$A \oplus B = \{x|[(\hat{B})_x \cap A] \subseteq A\} \tag{2.21}$$

[8]. An example of a dilation can be seen in Figure 2.9.

2.4.3 Erosion. The erosion of an image is a process that shrinks image A by the structuring element B. The process consists of finding the set of all points x where B shifted by x is contained in set A. This process is defined by

$$A \ominus B = \{x|(B)_x \subseteq A\} \tag{2.22}$$

[8]. A binary erosion can be seen in Figure 2.9.

2.4.4 Opening. Opening is a combination of an erosion followed by a dilation. The opening process generally smoothes the contour and eliminates thin lines. A opened by B is described by

$$A \circ B = (A \ominus B) \oplus B \tag{2.23}$$

[8]. An example is shown in Figure 2.9. This image illustrates how thin lines in the image are removed. The thin lines in the lower right corner and the small square in the upper right corner of the original image have been removed.

2.4.5 Closing. Closing is also a combination process. It consists of a dilation followed by an erosion. The closing process tends to smooth the contour,

fuse narrow breaks, and eliminate small holes in the image. A closed by B is described by

$$A \bullet B = (A \oplus B) \ominus B \tag{2.24}$$

[8]. An example is shown in Figure 2.9. This shows how the closing process can be used to eliminate small holes in the image that could be due to noise. Notice that the small white square that was near the center of the original image has been removed, but the larger square holes remained intact. This process is useful for removing small (smaller or equal to the size of the structuring element) bits of noise from images.

2.4.6 Gray-Scale. Until now, this section has dealt with binary images, but there are also equations governing the morphological processing of gray-scale images. The functions are similar to binary morphological functions, only another dimension has been added. Gray-scale dilation is described by

$$(A \oplus B)(s,t) = \max\{A(s-x,t-y)+B(x,y) | (s-x),(t-y) \in A; (x,y) \in B\} \tag{2.25}$$

and gray-scale erosion is described by

$$(A \ominus B)(s,t) = \min\{A(s+x,t+y)-B(x,y) | (s+x),(t+y) \in A; (x,y) \in B\} \tag{2.26}$$

[8].

Gray-scale dilation consists of shifting the structuring element to all positions in the image where at least one pixel overlaps, as shown in Figure 2.10. For each shift, the overlapping regions of the structuring element are added to the image and the maximum pixel value in that new region becomes the pixel value for the new dilated image. Gray-scale erosion is similar. It consists of shifting the structuring element to all positions in the image where the structuring element is entirely contained in the image as shown in Figure 2.10. For each shift, the structuring element is subtracted from the image and the minimum pixel value becomes the new eroded pixel value.

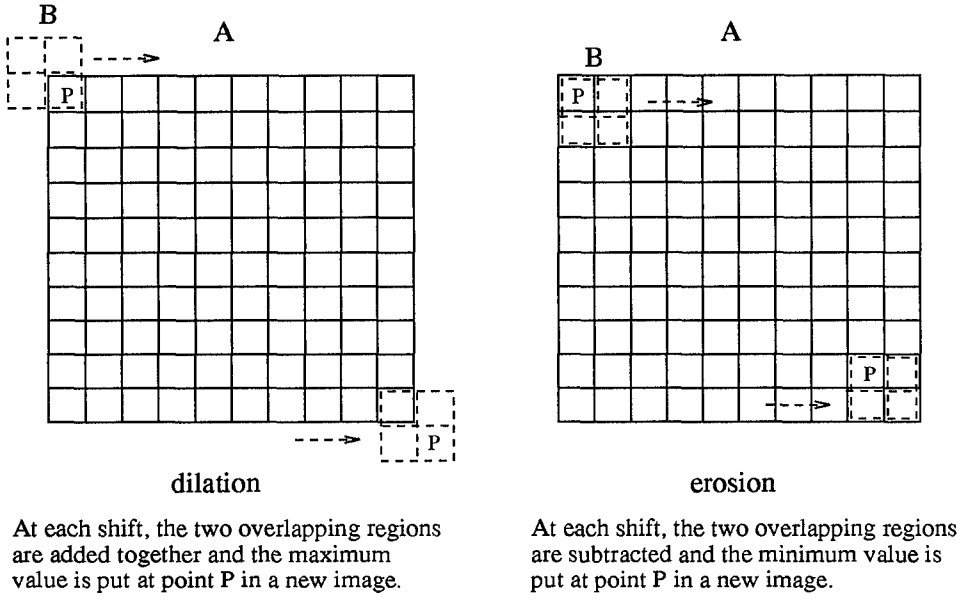


Figure 2.10 Gray-scale dilation and erosion.

Just as in binary morphological operations, the gray-scale opening of an image is formed by performing an erosion followed by a dilation of the eroded image. Similarly, the gray-scale closing is formed by performing a dilation followed by an erosion. These basic gray-scale morphological functions ultimately lead to the CMO (close minus open) algorithm. This CMO algorithm is used for clutter reduction in IR images by removing large area background regions and converting all objects in the scene to hot [1]. An example showing and describing each stage in the gray-scale CMO process can be seen in Chapter III.

2.5 Conclusion

This chapter described the basic theory behind the methods used in this research to process images and measure the classification errors incurred. The next chapter describes how this theory is implemented using both Khoros and Matlab software.

III. Methodology

Some of the image processing was done in Khoros, so a quick tutorial on Khoros is included in Appendix A. The rest of the code used in this research is included in Appendices B, C, and D. This chapter will describe the methods used to retrieve the images from the 8mm tapes, prepare them for processing, extract spatial features, perform the morphological transform, and find the errors incurred through classification.

3.1 Preparing the Images

The images were received on 8mm tape in ARF format from Wright Laboratories Automatic Target Recognition Branch. Along with the images came the code needed to transform the data from ARF to the VIFF format used in Khoros. This transformation is achieved by creating an arf2viff glyph in the Cantata workspace. This is done by selecting from the Cantata menu **workspace/file utilities/UIS filename/arf2viff.pane**. Now the images can be viewed, processed in Khoros, or saved into matrix files to be imported into Matlab.

3.2 Khoros Spatial Bands

The Khoros image processing environment has a built-in function to extract spatial features from images. These features are the mean, variance, contrast, angular 2^{nd} moment, entropy, and dispersion. The following equations that describe how each spatial feature is calculated were obtained from the Khoros manual by Donohoe [3]. The algorithm assumes there are K gray levels in the image. In this case, $K = 255$ and $p(k) =$ the probability of occurrence of each gray level k, where $k = 0, ..., 255$.

$$\text{mean, } \mu = E[k] = \sum_{k=0}^{K} k\, p(k) \tag{3.1}$$

$$\text{variance} = E[k^2] - E[k]^2 = \sum_{k=0}^{K} k^2\, p(k) - \left[\sum_{k=0}^{K} k\, p(k)\right]^2 \tag{3.2}$$

$$\text{contrast} = \sum_{k=0}^{K} k^2\, p(k) \tag{3.3}$$

$$\text{angular } 2^{\text{nd}} \text{ moment} = \sum_{k=0}^{K} p^2(k) \tag{3.4}$$

$$\text{entropy} = -\sum_{k=0}^{K} p(k) \log_2 p(k) \tag{3.5}$$

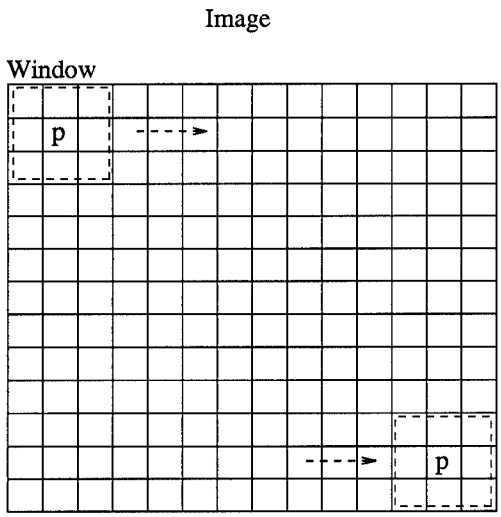
$$\text{dispersion} = \sum_{k=0}^{K} |k - \mu|\ p(k) \tag{3.6}$$

For each feature band, the window is shifted to be centered on every point in the original image where the window is fully contained in the image, as shown in Figure 3.1. The window size used in the calculations for this thesis is 3x3. Each of the six spatial feature extraction techniques are performed on the pixels within the windowed region, with the results becoming the new pixel values in the six extracted image bands. Due to the way the window is placed, there will be a one pixel border of zeros in the new bands. For more details on the spatial feature band extraction, see Appendix A. An actual example showing an original image along with the six spatial bands extracted from can be seen in Figures 3.2 and 3.3. (These bands have been scaled so the maximum gray level is 255.)

3.3 Morphology

The close minus open morphological transform that was described in Chapter II is performed on the image using the following Matlab script file and functions: cmo.m, closing.m, opening.m, erosion.m, and dilation.m. These are included in Appendix B.

The script file cmo.m loads the image data file and allows the user to define the morphological kernel, B. These two data matrices are then passed to the function



For each shift, the spatial algorithms are implemented and the results are put in the center point (p) of the window in the new image bands. (The 3x3 window produces a one pixel border of zeros in the new image bands.)

Figure 3.1 How the Khoros spatial bands are created.

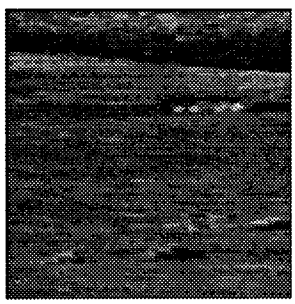


Figure 3.2 The original image.

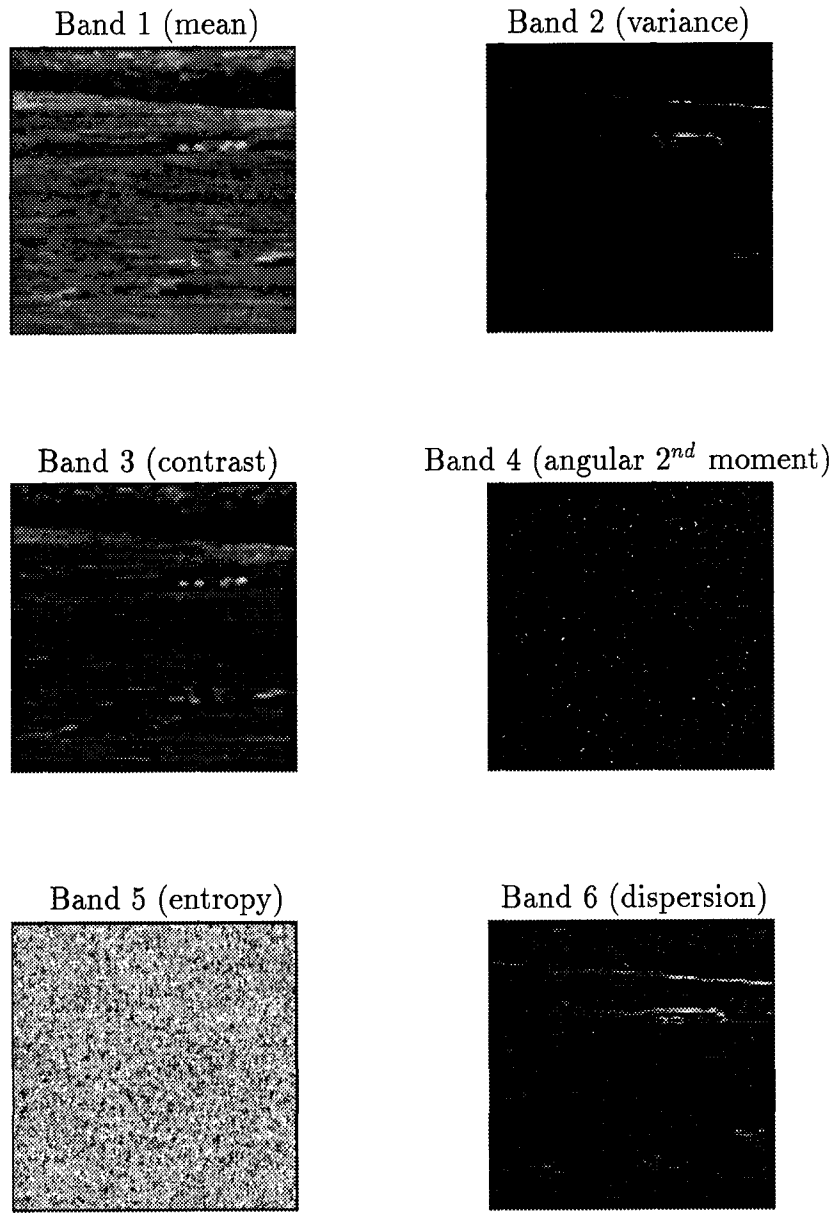


Figure 3.3 Khoros spatial band images.

closing.m which performs a dilation of the original image and then an erosion of the dilated image. Then the original two matrices are passed to opening.m which performs an erosion followed by a dilation. Finally, the opened image is subtracted from the closed image to complete the close minus open transform, and the resulting image is ready to be put into the classifying code. A step by step example using real images and showing each of these stages is shown in Figures 3.4 and 3.5.

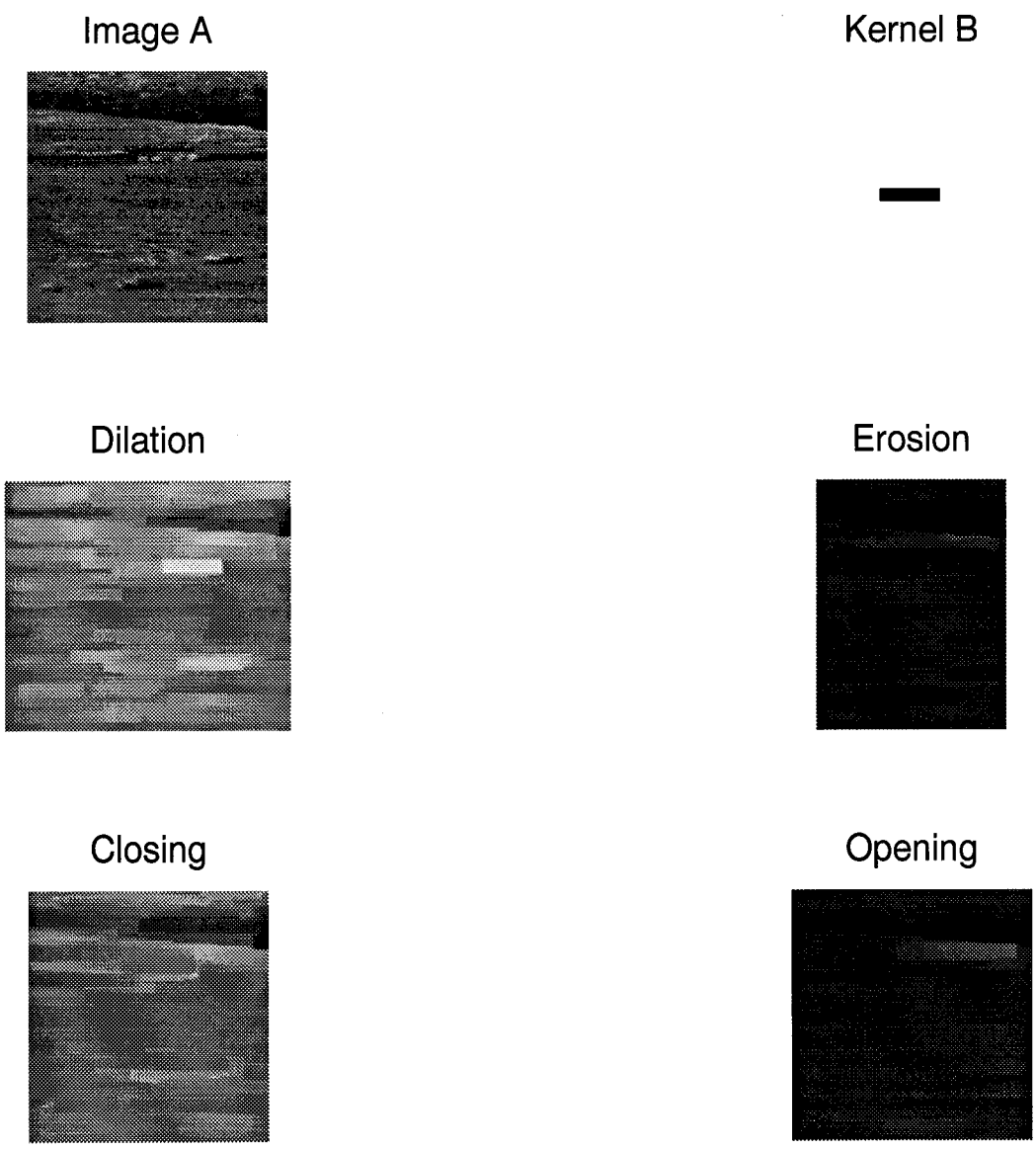


Figure 3.4 A step by step example of gray-scale morphology.

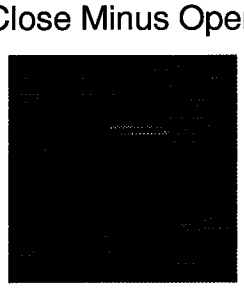


Figure 3.5 The output for the gray-scale close minus open function.

In Figure 3.4 the kernel is chosen to match the size of the target object (the scud in this case). The dilation expands the image and creates large bright areas where there were bright (hot) areas in the original image. In this case, there is a large bright area corresponding to the scud location. In just the opposite way, the erosion shrinks an image and creates large dark areas where there were dark (cold) spots in the original image.

Ideally, the result of a closing should be bright areas where there are bright targets in the image, while the result of the opening should be dark areas where there are dark targets in the image. In both cases, the backgrounds should be similar, so when the CMO is completed, the background will subtract away, leaving both hot and cold targets brighter than the new background level. Figure 3.5 shows how the scud location is brighter than the background. The locations of the small targets in the foreground are only slightly brighter than the areas around them. The CMO transform did not perform as well on the small targets because the kernel size was chosen to match the scud.

3.4 Classifying Code

The code used to find the error incurred through classification was obtained from Curtis Martin and modified to give the output information of interest to this thesis work [11].

Matlab script files are used to load the output of the various image processing techniques and put the data into a form suitable for use in the classifying code. These files pass a target matrix, a background matrix, the Parzen window size, and the number of nearest neighbors to be used in the classifying code. Both the target and background matrices must be created so that the columns represent separate data points (pixels in the case of the IR images) and the rows contain the features associated with those points. The two matrices do not have to be the same size, but they must contain the same number of rows (features). Martin's original classifying code is then used to determine the best values to input for the window size and the number of nearest neighbors.

3.4.1 Bayes Rule. In order to understand the calculation of the error probabilities, it is useful to note the decision method. The Bayes decision rule used in this code is a manipulation of Equation 2.4. It is called the *log-likelihood discriminant function* of x and is expressed by

$$l(x) = -\ln \frac{p(s_1|x)}{p(s_2|x)} \underset{s_2}{\overset{s_1}{\lessgtr}} 0 \tag{3.7}$$

[11]. The a posteriori probability $p(s_i|x)$ is

$$p(s_i|x) = \frac{p_i(x)P(s_i)}{p(x)} \tag{3.8}$$

where $P(s_i)$ is the a priori probability for class s_i, $p(x)$ is the pdf of x, and $p_i(x)$ is the class conditional pdf of x for class s_i. Substituting this into Equation 3.7, assuming equal a priori probabilities, replacing $p_i(x)$ with its estimate $\hat{p}_i(x)$, and introducing a threshold t to compensate for biases in the estimation, the result becomes

$$l(x) = -\ln \frac{\hat{p}_1(x)}{\hat{p}_2(x)} \underset{s_2}{\overset{s_1}{\lessgtr}} t \tag{3.9}$$

[7, 11].

3.4.2 Martin's Unmodified Code. The original unmodified code can be seen in Appendix C. In this original code, the hand-segmented target and non-target background sets, and a range of values for h and k are passed to the pknn.m function. Then the following procedure is performed five (the number of tests specified) times and the results are averaged together to obtain the final Bayes error estimate plots.

- For each set, every fifth point is taken as a test sample and the rest of the points are left over samples used to create the covariance matrices.
- The sample sets and inverse covariance matrices are passed to the function compute_distances.m which computes the squared Mahalanobis distances between all the points, both inter-class and intra-class. The distance matrices formed are then returned to pknn.m.
- Next, for each value of k, the density estimates and the resubstitution and leave-one-out discriminant values are calculated for the k-NN technique. From Equation 2.11 the k-NN density estimation is

$$\hat{p}_i(x_r^{(m)}) = \frac{k-1}{N_i V_d |\sum_i|^{1/2} \sqrt{[d_i^2(x_r^{(m)}, x_{k-NN}^{(i)})]^n}} \tag{3.10}$$

where $x_r^{(m)}$ is the rth sample of class s_m. When $i = m$, the resubstitution estimate is obtained by counting $x_r^{(m)}$ as its own nearest neighbor, while the leave-one-out estimate does not, but replaces N_i with $(N_i - 1)$ [11]. Substituting this into Equation 3.9, the discriminant function is

$$l(x_r^{(m)}) = \ln \frac{N_1 |\sum_1|^{1/2}}{N_2 |\sum_2|^{1/2}} + \frac{n}{2} \ln \frac{d_1^2(x_r^{(m)}, x_{k-NN}^{(1)})}{d_2^2(x_r^{(m)}, x_{k-NN}^{(2)})}. \tag{3.11}$$

These values are passed to classify.m which calls min_error.m to find the resubstitution minimum error and threshold. This is done by determining how the two discriminant value sets overlap, varying a threshold over that region, and counting the errors (misclassifications). The resubstitution minimum er-

ror and threshold values are returned to classify.m and then the leave-one-out minimum error and threshold are calculated in a similar fashion. These error values are then returned to pknn.m.

- The same procedure is followed for the Parzen window technique for each value of h, but using different equations for the density estimation and the discriminant values. The resubstitution density estimate is

$$\hat{p}(x_r^{(m)}) = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{1}{h^n} k_i \left(\frac{x_r^{(m)} - x_j^{(i)}}{h} \right) \tag{3.12}$$

while the leave-one-out estimate is

$$\hat{p}(x_r^{(m)}) = \frac{1}{N_i - 1} \left[\sum_{j=1}^{N_i} \frac{1}{h^n} k_i \left(\frac{x_r^{(m)} - x_j^{(i)}}{h} \right) - \frac{k_i(0)}{h^n} \right]. \tag{3.13}$$

Substituting these into Equation 3.9, using a Gaussian window, and manipulating the equations, the resulting likelihood ratios for the resubstitution (R) method and leave-one-out (L) methods for class s_1 and s_2, respectively, are

$$l_R(x_r^{(m)}) = \ln \frac{N_1 |\sum_1|^{1/2}}{N_2 |\sum_2|^{1/2}} - \ln \frac{\sum_{j=1}^{N_1} exp\left(-d_1^2(x_r^{(m)}, x_j^{(1)})/2h^2\right)}{\sum_{j=1}^{N_2} exp\left(-d_2^2(x_r^{(m)}, x_j^{(2)})/2h^2\right)} \tag{3.14}$$

$$l_L(x_j^{(1)}) = \ln \frac{(N_1 - 1) |\sum_1|^{1/2}}{N_2 |\sum_2|^{1/2}} - \ln \frac{\sum_{j=1}^{N_1} exp\left(-d_1^2(x_j^{(1)}, x_j^{(1)})/2h^2\right) - 1}{\sum_{j=1}^{N_2} exp\left(-d_2^2(x_j^{(1)}, x_j^{(2)})/2h^2\right)} \tag{3.15}$$

$$l_L(x_j^{(2)}) = \ln \frac{N_1 |\sum_1|^{1/2}}{(N_2 - 1) |\sum_2|^{1/2}} - \ln \frac{\sum_{j=1}^{N_1} exp\left(-d_1^2(x_j^{(2)}, x_j^{(1)})/2h^2\right)}{\sum_{j=1}^{N_2} exp\left(-d_2^2(x_j^{(2)}, x_j^{(2)})/2h^2\right) - 1}. \tag{3.16}$$

For a complete description of the steps involved, see the work by Martin [11].

- All of the error values are then passed back to the driver script file where plots can be created.

3.4.3 Modified Code. The modified code used in creating the ROC curves can be seen in Appendix D. The procedure used in the modified code is very similar

to that in the original code. The values of h and k that gave the best results for the leave-one out method in the original code are the ones used in the modified code. These values, along with the hand-segmented target and background sets, are passed to the pknn2.m function. The procedure is the same as before except that instead of finding the minimum error and threshold, the min_error2.m function determines where the two discriminant sets overlap and varies a threshold over the overlapping region. For each threshold value, the error probabilities are calculated using the following rule: the number of set 1 discriminant values below the threshold divided by the total number of set 1 discriminant values is the probability of a hit, and the number of set 2 discriminant values below the threshold divided by the total number of set 2 discriminant values is the probability of false alarm. These probabilities are then returned to the driver script where they can be plotted to obtain the final ROC curves desired.

Because the data is finite, the probabilities are discrete and the ROC curves created from them are not smooth. Each of the ten tests creates ROC curves of the form in Figure 3.6. Therefore, there needs to be a way to average these curves together to obtain an average ROC curve for each image processing technique. The function that does this is interpolate.m, which can also be seen in Appendix D. It makes the curves monotonic increasing and then uses the Matlab function interp1 to interpolate between the points, as shown in Figure 3.6. Then, once all ten curves have a hit probability value at each probability of false alarm, they can be averaged together to create the final ROC curve.

Due to the averaging involved, a confidence interval is also needed for the ROC curve. In this case, the standard deviation σ is unknown, but an estimate s can be calculated from the data. The distribution that can be used to compute limits on the average value is the *Student's t distribution* published by W.S. Gossett and perfected

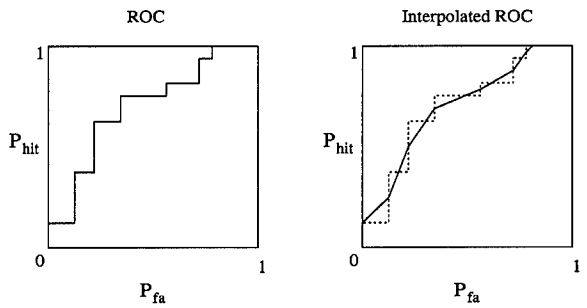


Figure 3.6 Example showing ROC curve and the interpolated ROC curve used in averaging.

by R.A. Fisher [2, 9, 17]. The confidence interval is given by

$$\left(\hat{p} - t_{.05}\frac{s}{\sqrt{n}}, \hat{p} + t_{.05}\frac{s}{\sqrt{n}}\right) \tag{3.17}$$

where $\hat{p}$ is the average hit probability, s is the standard deviation of the n samples used, and $t_{.05} = 2.262$ for a 95% confidence interval with $(n-1) = 9$ degrees of freedom [17].

3.5 Summary

This chapter described the methods used to process the images and create ROC curves. The next chapter displays and describes the results obtained from these methods.

IV. Results

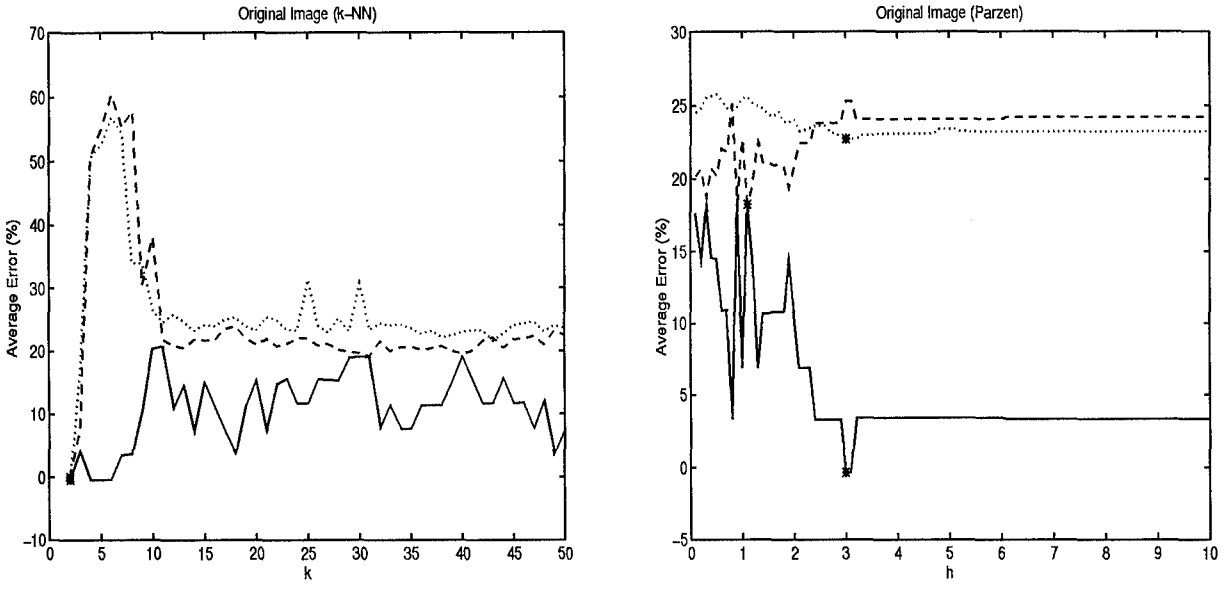
4.1 Preliminary Results

In order to create the ROC curves, values need to be chosen for both h and k. This is done using the original unmodified code obtained from Martin [11]. The output plots of this code show the total average error for resubstitution and the two leave-one-out methods for varying values of h and k, as shown in the example output in Figure 4.1. Here, the average error represents the percentage of total misclassifications, both false positives (false alarms) and false negatives (misses). The two leave-one-out methods, described in detail in Martin's thesis, are produced by two different threshold selection techniques [11]. Since the best values of h and k are found by trial and error, the h and k values that gave the smallest leave-one-out (option 2) error in the output of the original code are used as inputs to the modified code. The h and k values with the next smallest leave-one-out errors are used as needed. In this work, the leave-one-out option 2 is used because it is less biased than option 1. A compilation of the values of h and k that are actually used in the construction of the ROC curves can be seen in the tables in each section.

4.2 Test Problems

The test problems consist of finding ROC curves for the test data sets, whose actual distributions are shown in Figures 4.2 and 4.4.

For Test 1, both data sets are Gaussian distributions of points along the x axis. In the figure, they are offset in the y direction for illustration purposes only. The target set has zero mean and unit variance, and the non-target set has mean $\mu = 2$ and unit variance. Due to the known properties of this data, points on the empirical ROC curve can be found by varying the threshold along the x axis. The ROC curves for Test 1 can be seen in Figure 4.3. The values of h and k used to



(- Resubstitution, -- Leave-one-out 1, .. Leave-one-out 2)

Figure 4.1 The output for the original image using Martin's unmodified code.

create the ROC curves are shown in Table 4.1, and the areas under the leave-one-out curves are shown in Table 4.2.

Input Data	h	k
Test 1	3	15
Test 2	2	35

Table 4.1 Values chosen for h and k for the test sets.

For Test 2, the target set has zero mean and unit variance, and the background set has a mean along the x axis of $\mu = 3$ and unit variance. This is shown in Figure 4.4. The data is arranged so the empirical ROC can be found by varying a threshold along the x axis. The ROC curves for Test 2 can be seen in Figure 4.5. The h and k values used are shown in Table 4.1, and the areas under the leave-one-out curves are shown in Table 4.2.

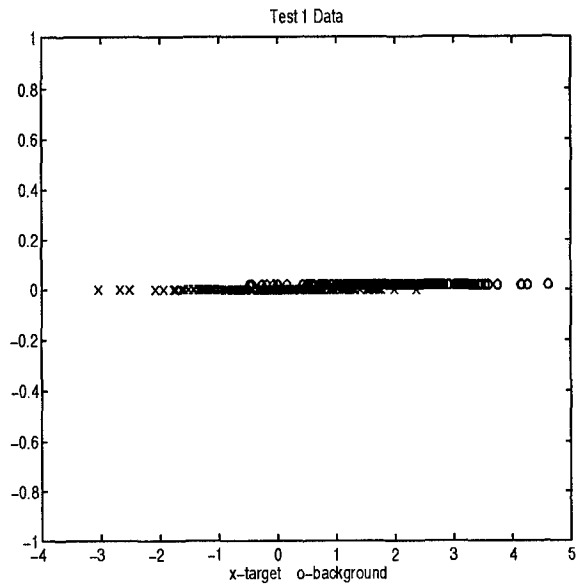


Figure 4.2 Data sets for test 1.

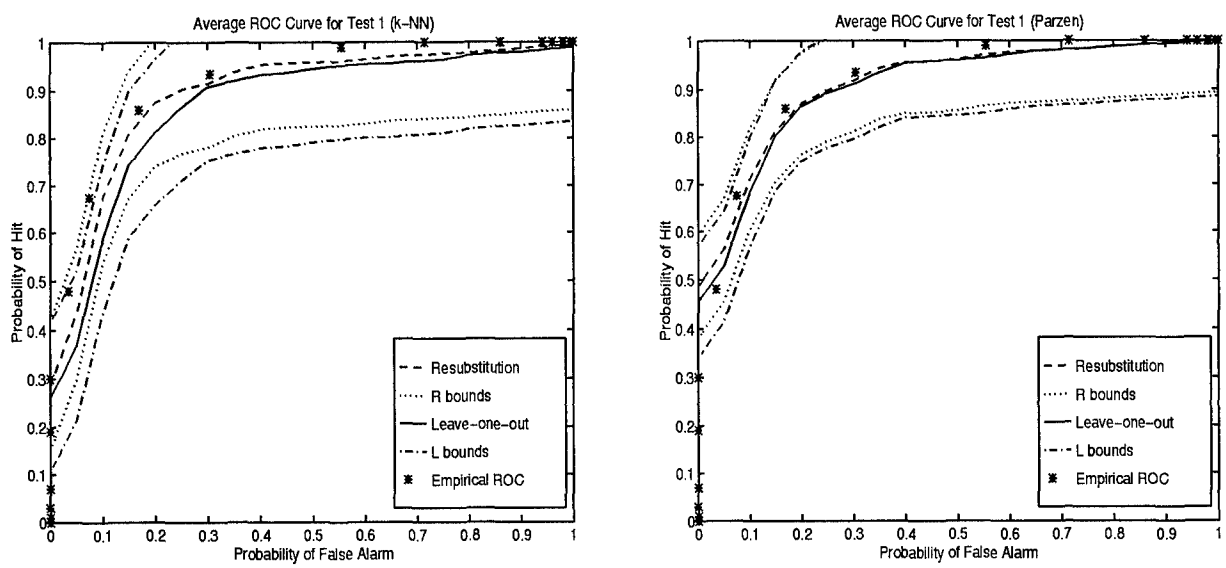


Figure 4.3 ROC curves for test 1.

From the figures for both tests, it can be seen that the resubstitution and leave-one-out estimates are good approximations to the empirical ROC curves for the test data.

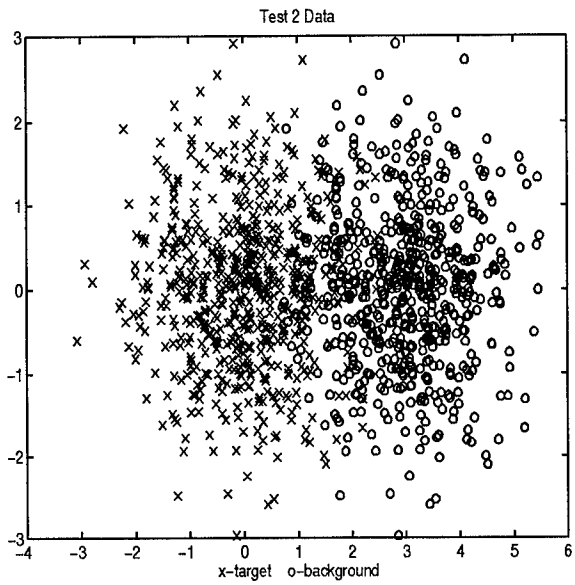


Figure 4.4 Data sets for test 2.

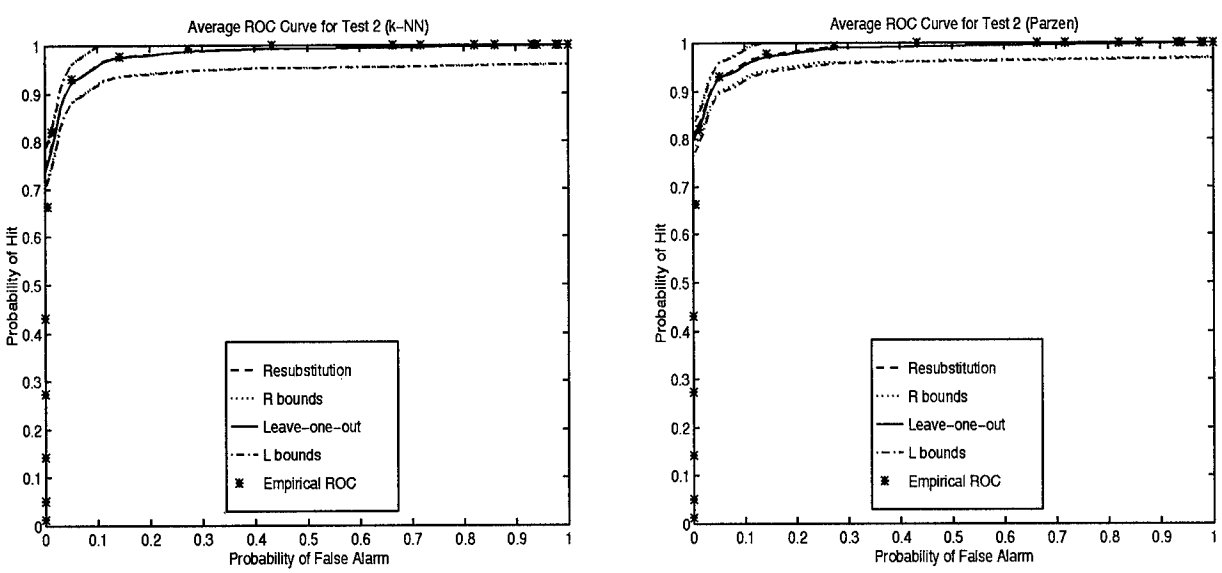


Figure 4.5 ROC curves for test 2.

Input Data	ROC Area (k-NN)	ROC Area (Parzen)
Test 1	.8652	.9020
Test 2	.9805	.9820

Table 4.2 Areas under the ROC for the test data sets.

4.3 Military Target Image (Targets v. Background)

The original IR image used in this section can be seen back in Figure 1.2. The actual values used for h and k are listed in Table 4.3.

Input Image	h	k
Original Image	3	20
CMO Morphological Filtered Image	0.1	22
Band 1 (mean)	2	32
Band 2 (variance)	9.7	37
Band 3 (contrast)	6.5	34
Band 4 (angular 2^{nd} moment)	3	35
Band 5 (entropy)	3	35
Band 6 (dispersion)	0.2	9
All 6 bands	0.3	26
Bands 1, 2, 3, & 6	1	20

Table 4.3 Values chosen for h and k for the target image of Figure 1.2.

4.3.1 Original Image. The ROC curves created using target and background matrices obtained from the original image are shown in Figure 4.6. These curves show the probabilities of hit pixels versus false alarm pixels.

4.3.2 Spatial Bands. The ROC curves obtained for the six separate spatial feature bands can be seen in Figures 4.7 through 4.12. The ROC curves for the combinations of the spatial feature bands can be seen in Figures 4.13 and 4.14. As can be seen in the figures and table, each spatial band on its own does not outperform the original unprocessed image, the exception being band 1. But when the bands are combined, the resulting ROC curves are better then the original ROC. Therefore, combining the spatial bands will produce a better segmentation technique.

4.3.3 Morphological Filter. The CMO morphological filter in this section uses a 10x50 pixel size kernel. The ROC curves in Figure 4.15 show that this seg-

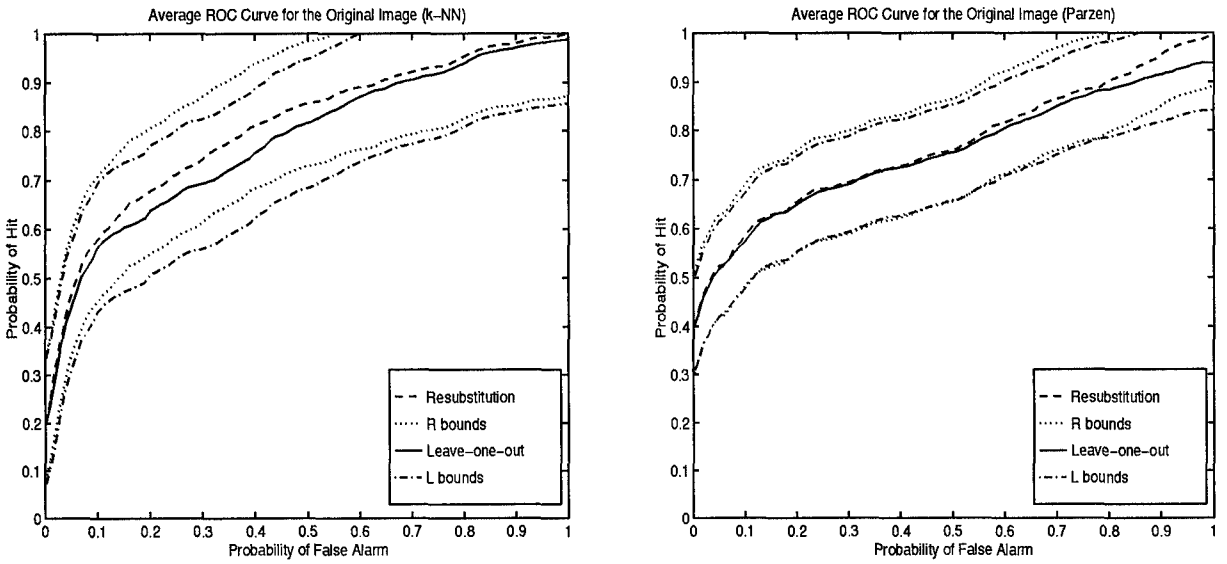


Figure 4.6 ROC curves for the original image.

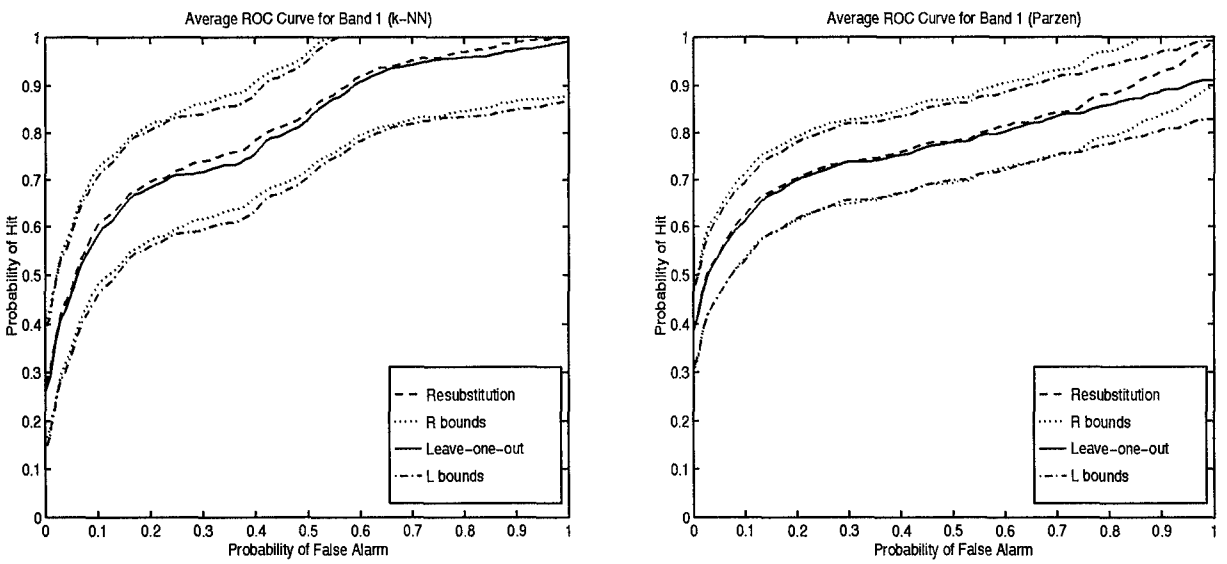
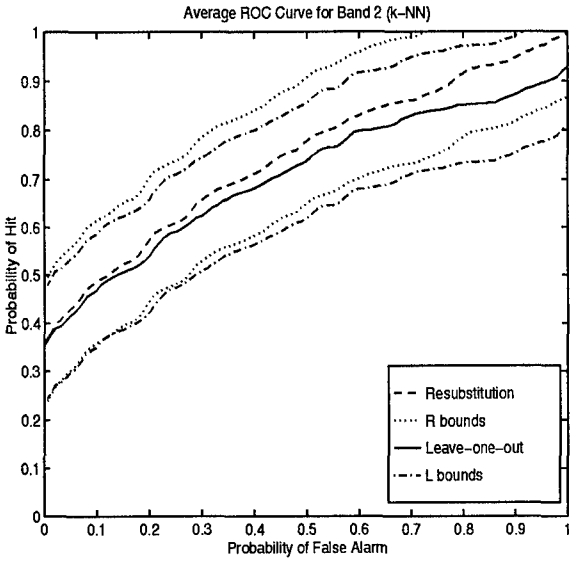


Figure 4.7 ROC curves for band 1 (mean).



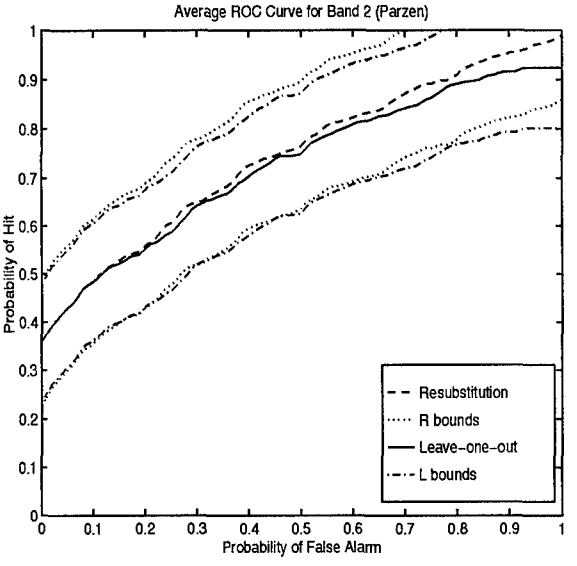
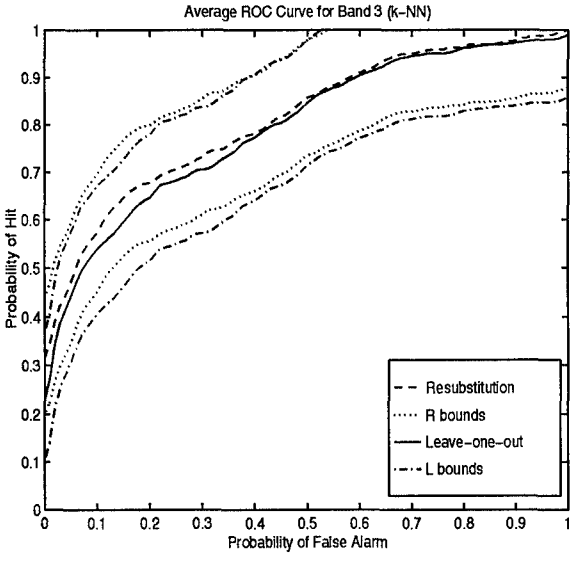


Figure 4.8 ROC curves for band 2 (variance).



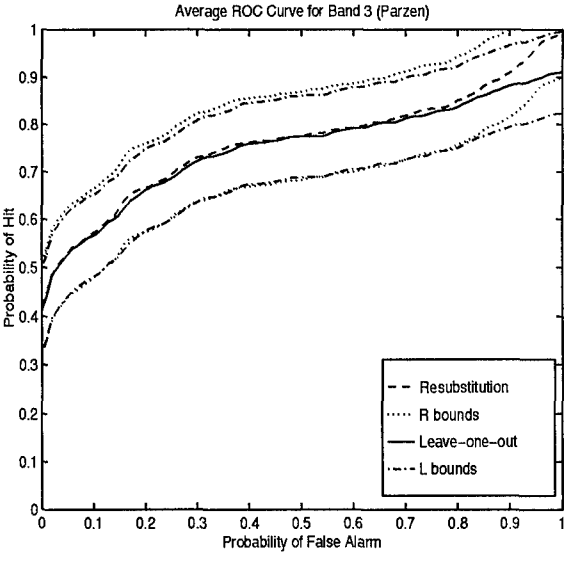


Figure 4.9 ROC curves for band 3 (contrast).

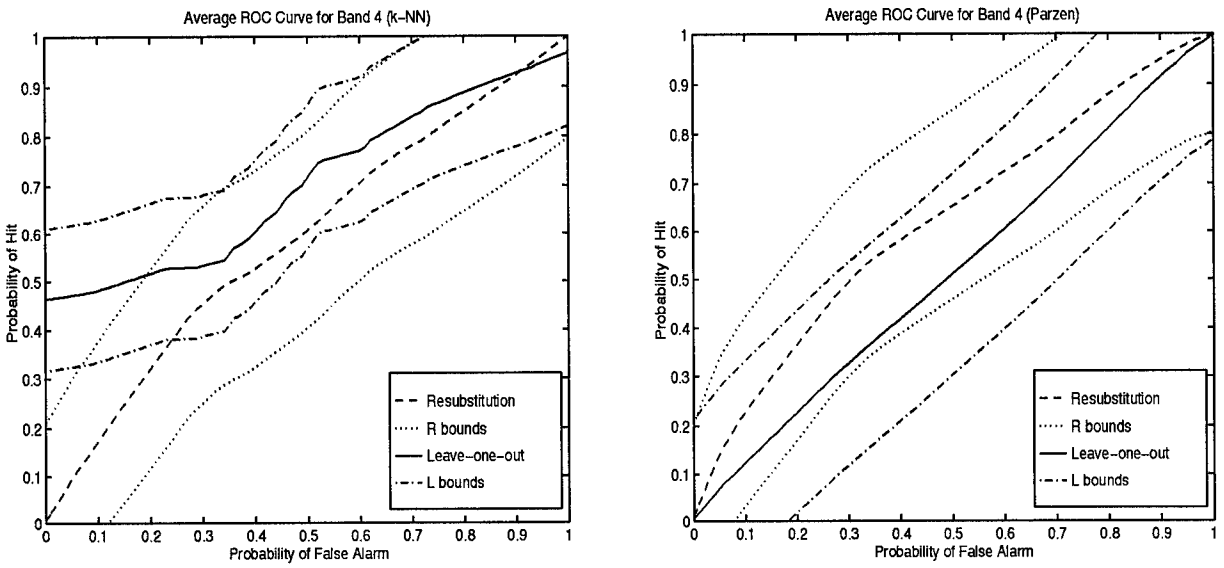


Figure 4.10 ROC curves for band 4 (angular 2^{nd} moment).

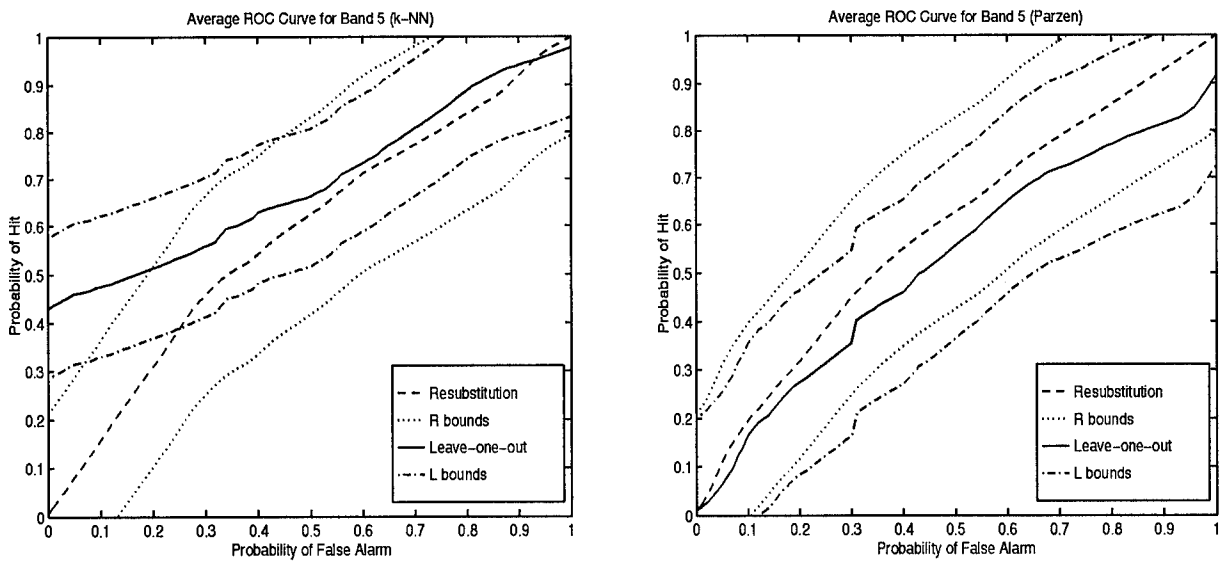


Figure 4.11 ROC curves for band 5 (entropy).

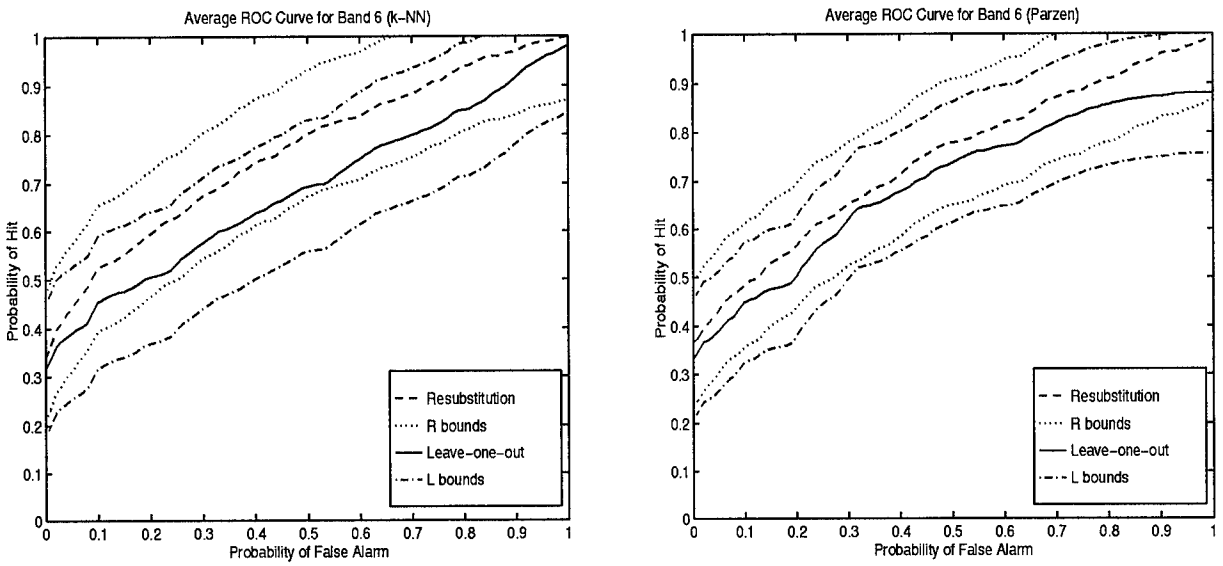


Figure 4.12 ROC curves for band 6 (dispersion).

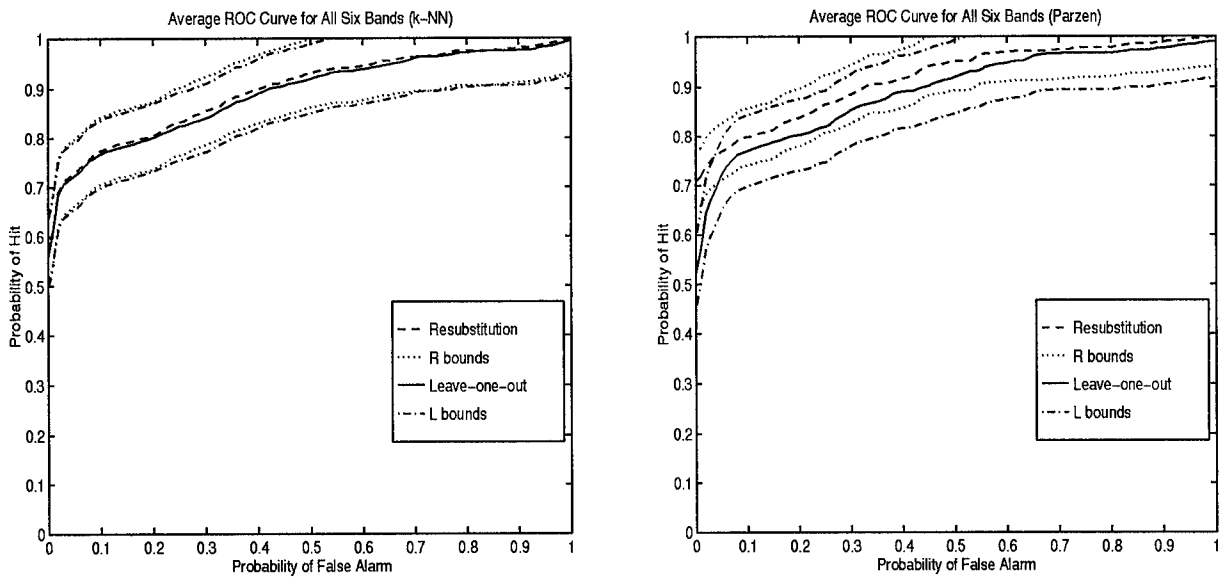


Figure 4.13 ROC curves for all six bands.

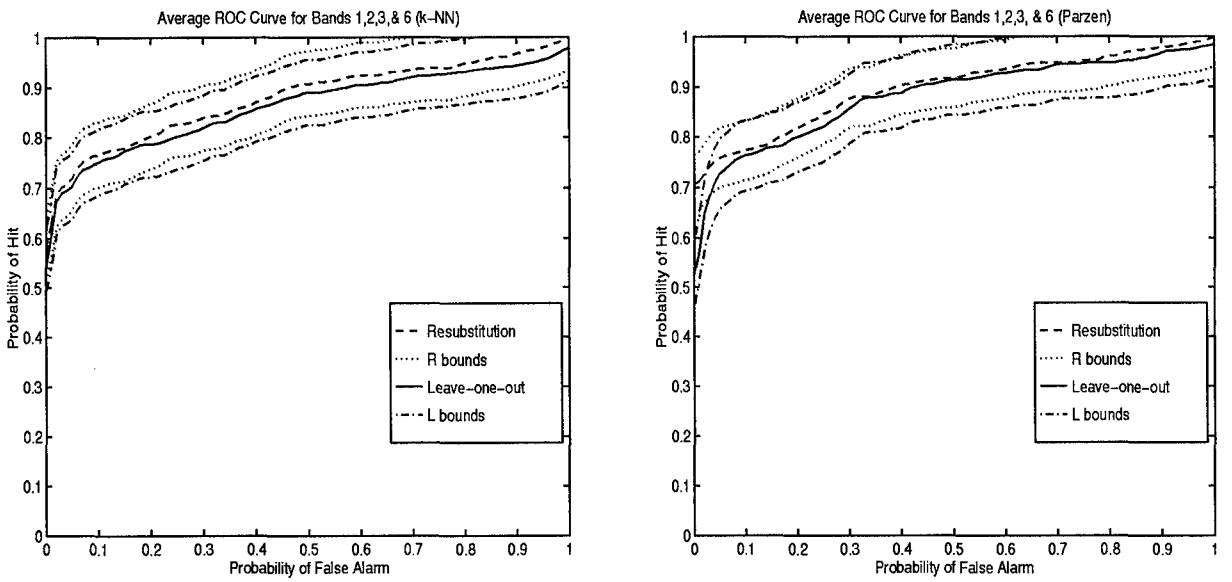


Figure 4.14 ROC curves for bands 1, 2, 3, and 6.

mentation technique performs better than the original unprocessed image as well as the spatial bands and the combined spatial bands. This is the expected result, considering the output processed images are of the types in Figures 3.3-3.5.

4.3.4 Section Summary. The ROC curves produced in this section show the performance of the image processing techniques. Of the techniques used here, the morphological CMO algorithm provided the best results, followed by the combinations of spatial feature bands. These are qualified as being better because the ROC curve is higher than those of the unprocessed image and the single spatial band features. Another way to measure this is by calculating the area under the ROC curve. These areas give numbers that can be compared. In many cases curves with larger areas are better, but that is not always the case. Two curves with equal areas may look entirely different, so which one is better depends upon the specific application. The areas under the average leave-one-out curves have been calculated and tabulated in Table 4.4.

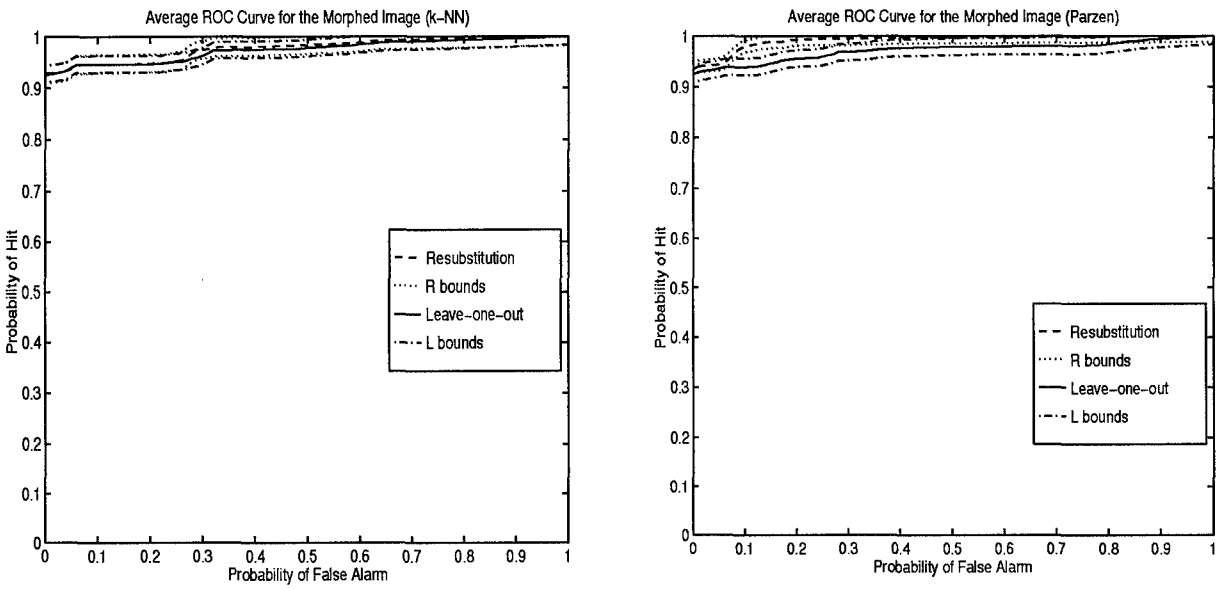


Figure 4.15 ROC curves for the 10x50 morphological CMO image.

Input Image	ROC Area (k-NN)	ROC Area (Parzen)
Original Image	.7770	.7546
CMO Morphological Filtered Image	.9744	.9716
Band 1 (mean)	.8006	.7645
Band 2 (variance)	.7050	.7239
Band 3 (contrast)	.7931	.7480
Band 4 (angular 2^{nd} moment)	.7001	.5152
Band 5 (entropy)	.6920	.5223
Band 6 (dispersion)	.6822	.6920
All 6 bands	.8881	.8886
Bands 1, 2, 3, & 6	.8600	.8798

Table 4.4 Areas under the leave-one-out ROC curves.

4.4 Scud Image (Scud v. Background and Other Targets)

The image from Figure 1.2 is also used in this section, but instead of identifying target versus non-target pixels, scud pixels are identified versus non-scud pixels. In this case, the other targets (tanks, trucks, etc.) are considered to be in the non-scud pixel set. The h and k values used can be seen in Table 4.5.

Input Image	h	k
Original Image	3	15
CMO Morphological Filtered Image	1	20
Band 1 (mean)	1	5
Band 2 (variance)	7	15
Band 3 (contrast)	1	15
Band 4 (angular 2^{nd} moment)	3	15
Band 5 (entropy)	3	15
Band 6 (dispersion)	2	20
All 6 bands	1	5
Bands 1, 2, 3, & 6	1	5

Table 4.5 Values chosen for h and k for the scud image of Figure 2.

4.4.1 Original Image. The ROC curves created using scud and non-scud matrices obtained from the original image are shown in Figure 4.16. Comparing Figure 4.16 with Figure 4.6, the scud versus non-scud pixel classification did not perform as well as the target versus non-target. This happens because the other target pixels in the image are similar to scud pixels, or at least more similar to the scud pixels than the background pixels.

4.4.2 Spatial Bands. The ROC curves obtained for the six separate spatial feature bands can be seen in Figures 4.17 through 4.22. The ROC curves for the combinations of the spatial feature bands can be seen in Figures 4.23 and 4.24.

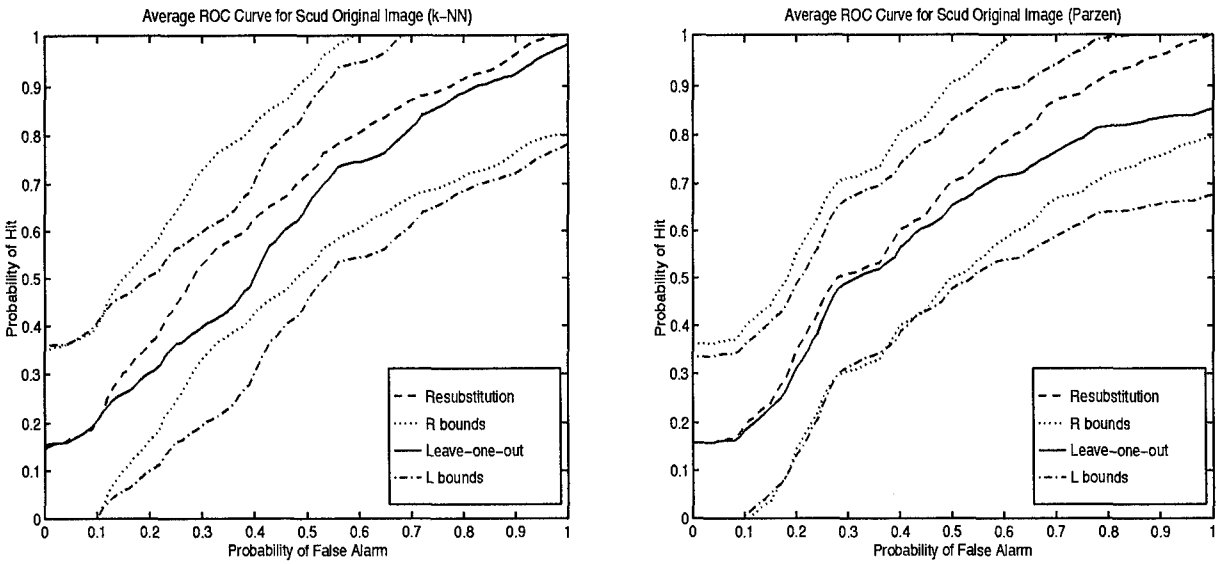


Figure 4.16 ROC curves for the scud original image.

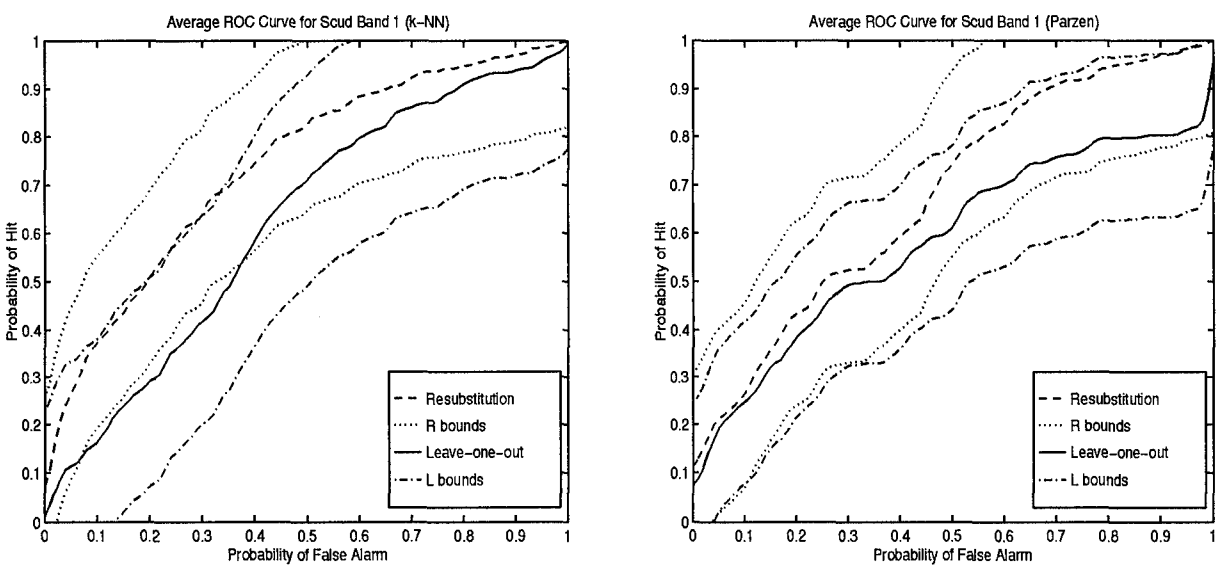


Figure 4.17 ROC curves for the scud band 1 (mean).

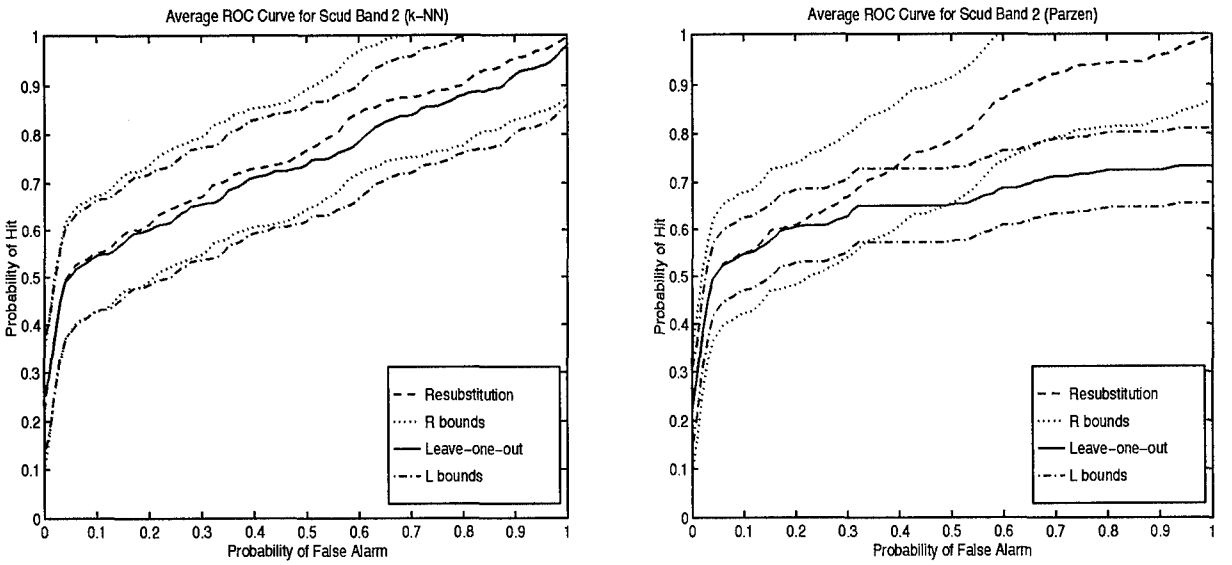


Figure 4.18 ROC curves for the scud band 2 (variance).

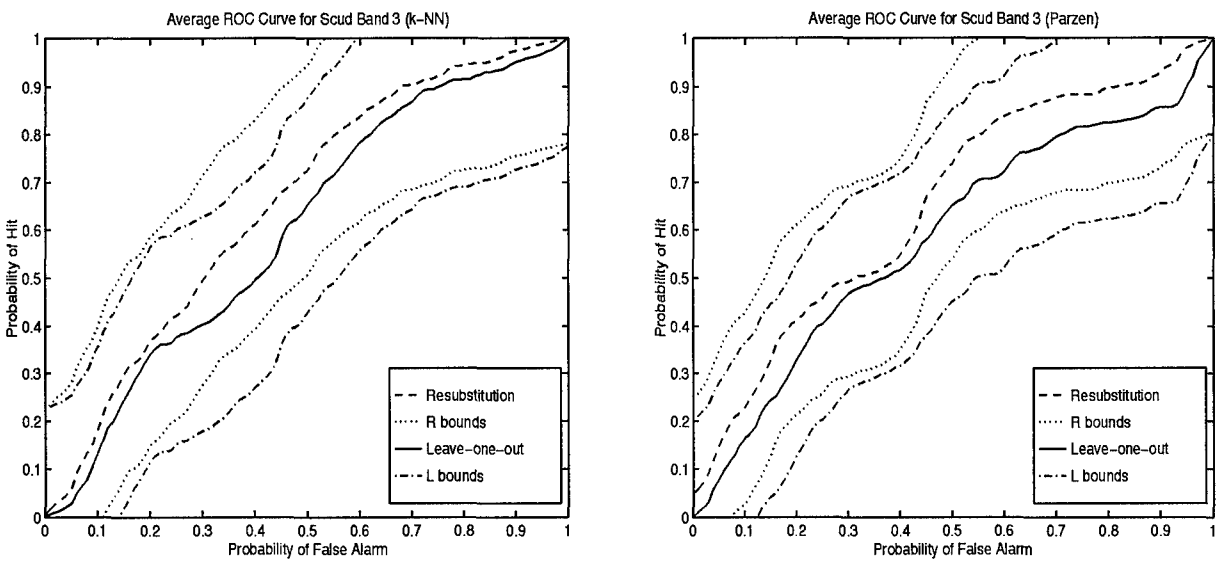


Figure 4.19 ROC curves for the scud band 3 (contrast).

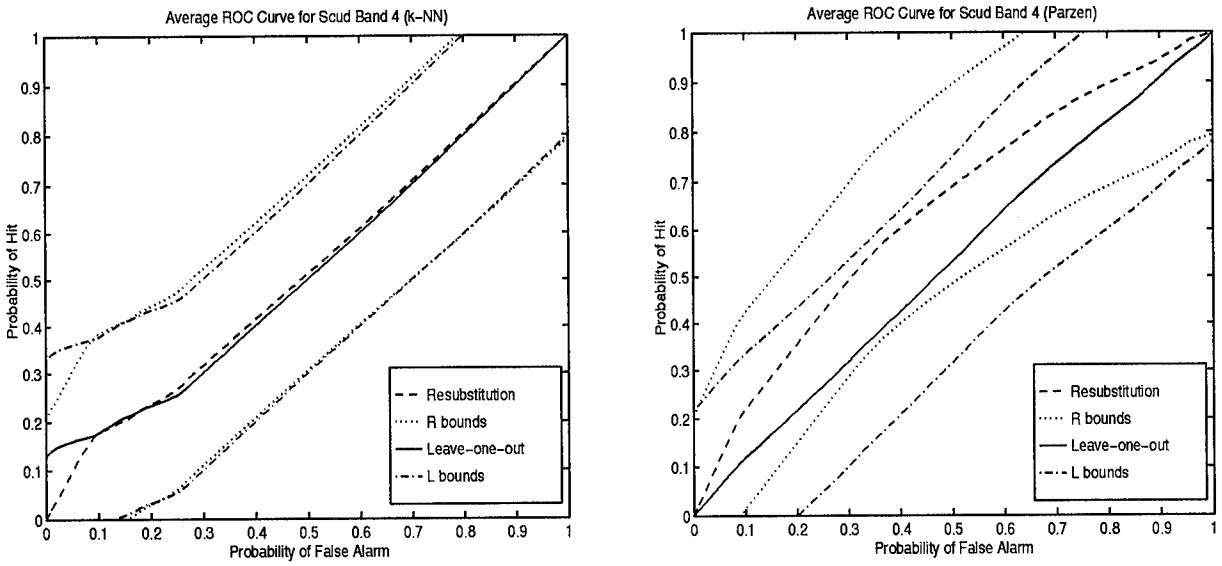


Figure 4.20 ROC curves for the scud band 4 (angular 2^{nd} moment).

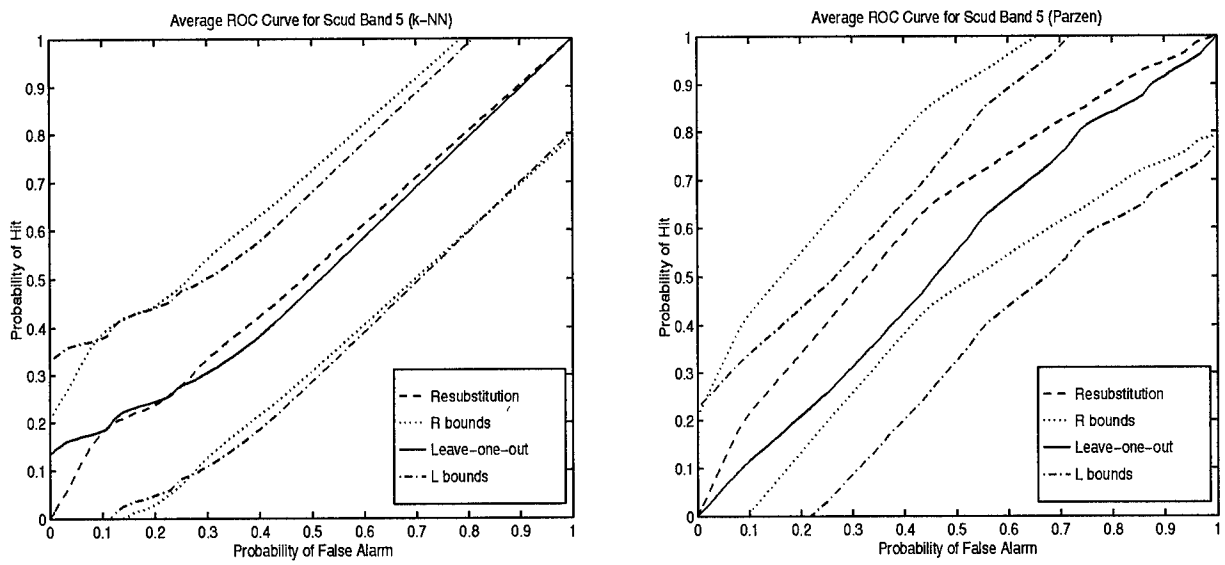


Figure 4.21 ROC curves for the scud band 5 (entropy).

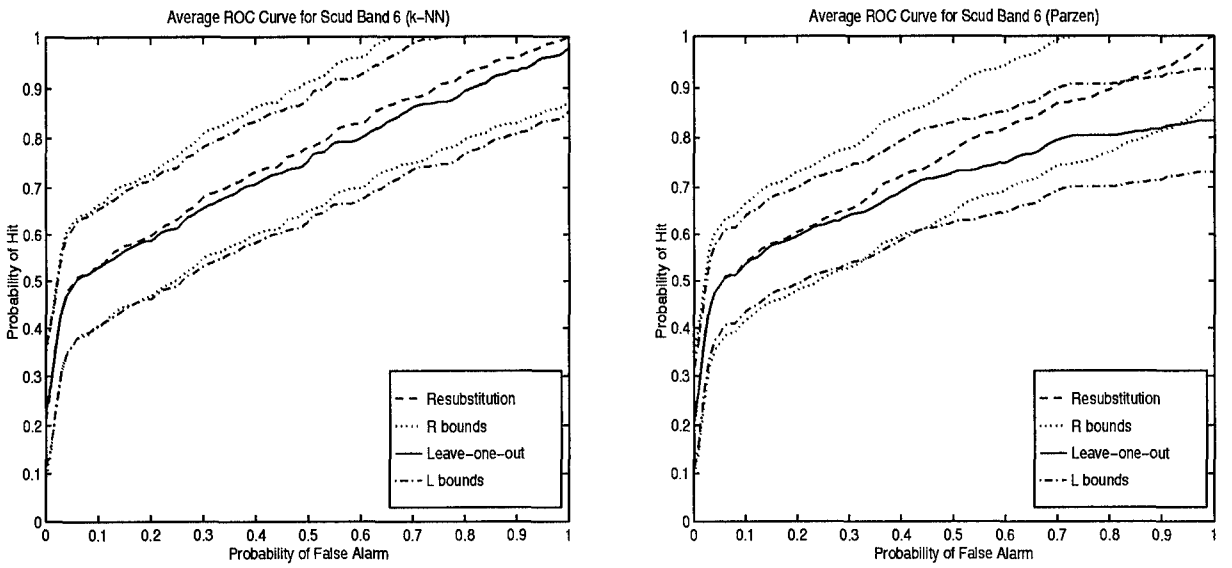


Figure 4.22 ROC curves for the scud band 6 (dispersion).

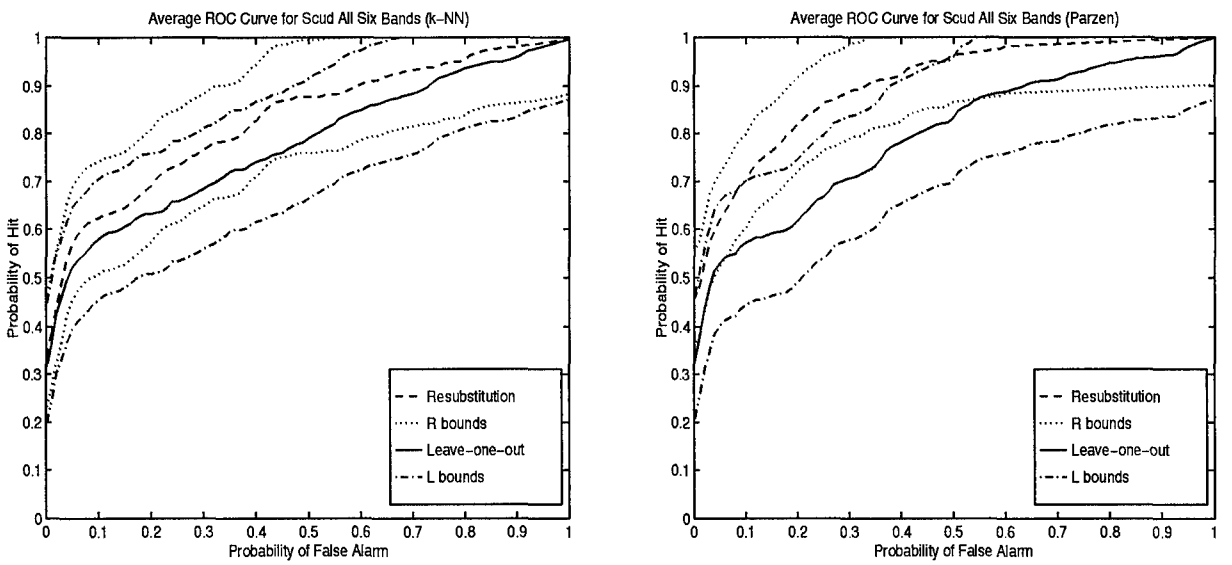


Figure 4.23 ROC curves for all six bands for the scud.

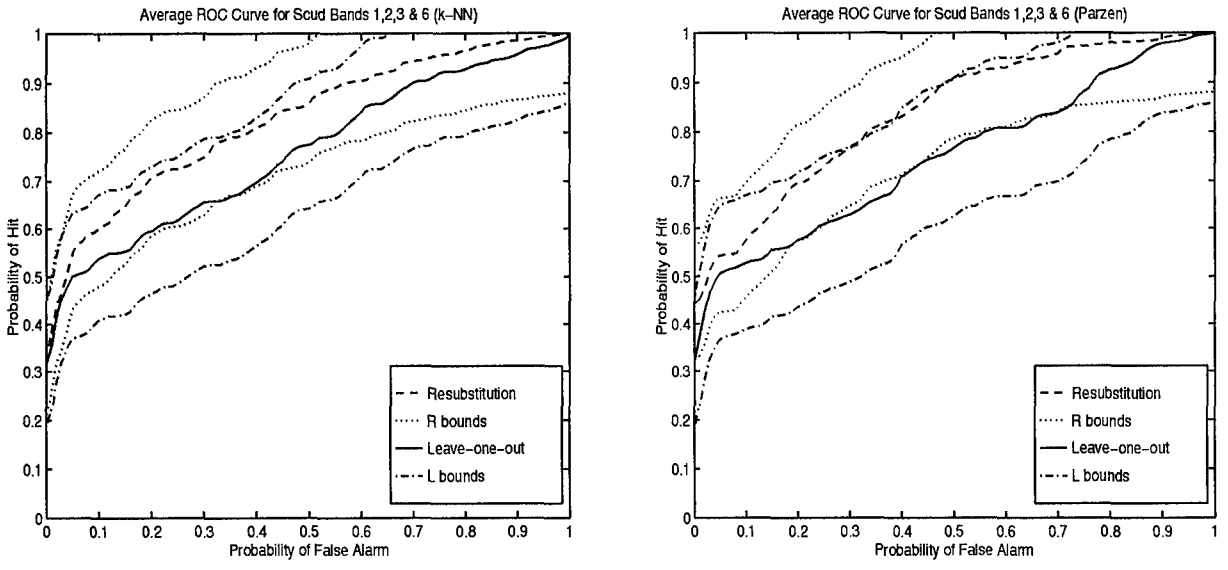


Figure 4.24 ROC curves for bands 1, 2, 3, and 6 for the scud.

These again show that combining the bands seems to give better results than using each band alone.

4.4.3 Morphological Filter. The ROC curves for the CMO morphological filtered scud image can be seen in Figure 4.25. Just as for the target versus non-target section, the ROCs created from the morphological processed image are better than those of the original and the spatial bands. Even so, these ROC curves are not as good as those for the target versus background for the same reason as stated before.

4.4.4 Section Summary. The ROC curves produced in this section show the performance of the image processing techniques when used to find only the scud. Of the techniques used here, the morphological CMO algorithm provided the best results, followed by the combinations of spatial feature bands, just as in section 4.3 for the target versus non-target problem. The areas under the average leave-one-out curves have been calculated and tabulated in Table 4.6.

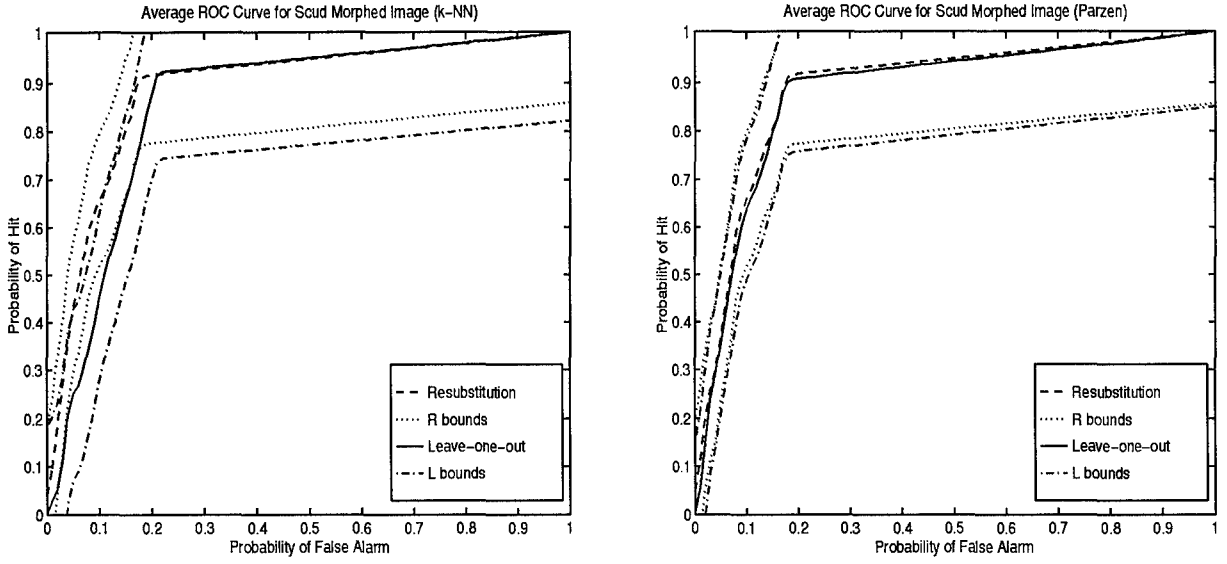


Figure 4.25 ROC curves for the 10x50 morphological CMO scud image.

Input Image	ROC Area (k-NN)	ROC Area (Parzen)
Original Image	.6015	.5811
CMO Morphological Filtered Image	.8528	.8731
Band 1 (mean)	.6188	.5814
Band 2 (variance)	.7337	.6473
Band 3 (contrast)	.6021	.5805
Band 4 (angular 2^{nd} moment)	.5187	.5215
Band 5 (entropy)	.5144	.5289
Band 6 (dispersion)	.7393	.6951
All 6 bands	.7756	.7936
Bands 1, 2, 3, & 6	.7561	.7459

Table 4.6 Areas under the leave-one-out ROC curves.

4.5 Breast Cancer Images (Cancerous v. Healthy)

The methods developed to estimate the hit and false alarm probabilities for the military target problem can also be applied to other areas, such as the breast cancer problem. The breast cancer images used in this research were provided by The University of Cincinnati. An example image can be seen in Figure 4.26.

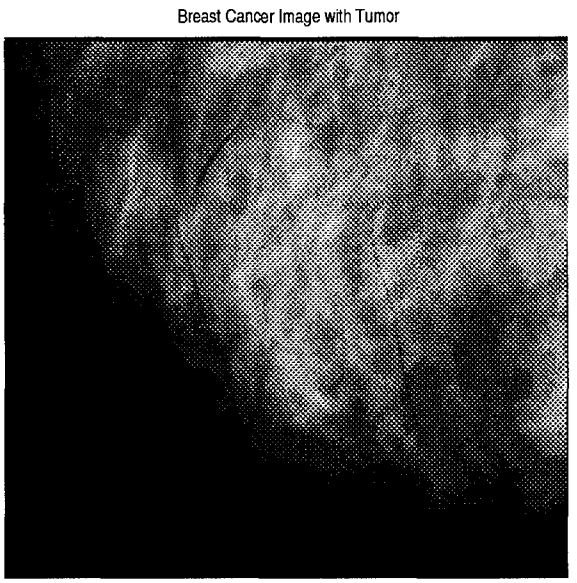


Figure 4.26 A breast cancer image with a tumor.

A part of the image within the marked area is used as the target set, and a section from the upper right corner (outside the marked area) is used as the background set. These signify tumor and healthy tissue, respectively. The resulting ROC curves for $h = 1$ and $k = 15$ are displayed in Figure 4.27. The areas under the average leave-one-out ROC curves for the k-NN and Parzen methods are .8531 and .8084, respectively.

4.6 Breast Cancer Images (Malignant v. Benign)

As well as finding tumors, the code is used to find the probabilities associated with finding malignant versus benign image pixels in the tumors. Examples of

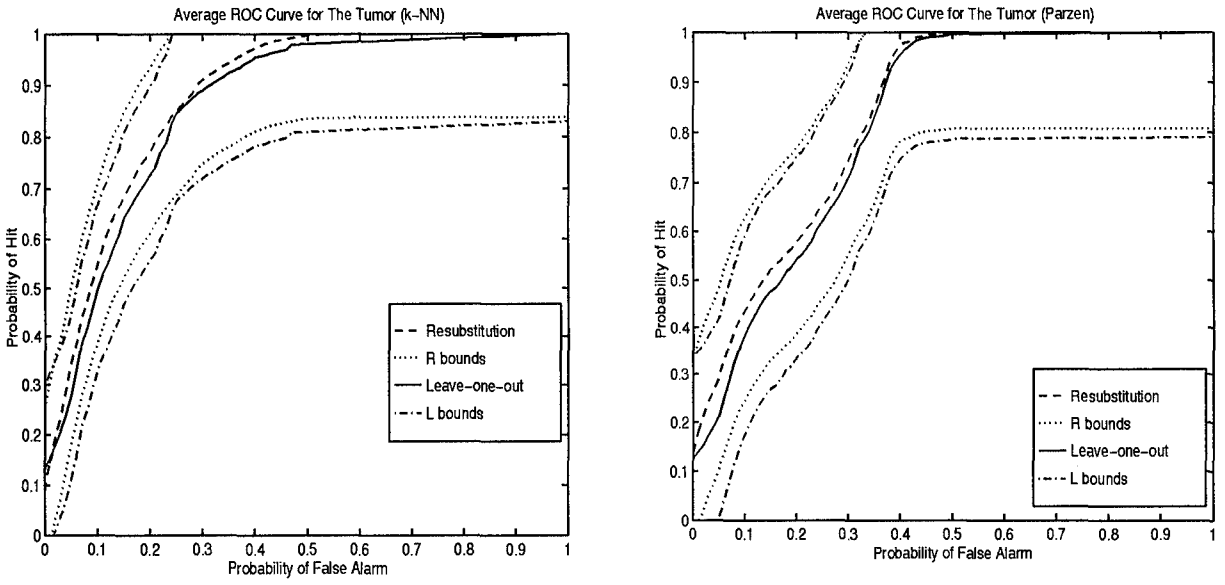


Figure 4.27 ROC curves for cancerous versus healthy tissue.

truthed pieces of the images showing malignant and benign tumors can be seen in Figure 4.28.

Following the ordering from left to right in Figure 4.28, each pair of malignant and benign tumors are used as target and non-target input sets to the ROC code. The leave-one-out ROCs for each trial are shown in Figure 4.29, and a ROC for all six pairs combined is shown in Figure 4.30. Table 4.7 shows the values used for h and k for each of the six pairs of images.

The ROC curves produced in this section show the ability of the algorithm to determine the probability of distinguishing between malignant and benign tumor pixels. As in previous sections, the areas under the ROC curves have been calculated and are listed in Table 4.8.

Next, the same six pairs of malignant and benign tumors are put through the angular 2^{nd} moment spatial feature extraction in Khoros and the ROC analysis is performed. The h and k values used to create the ROC curves are listed in Table 4.9. The leave-one-out ROC curves for each pair are shown in Figure 4.31 and the ROC

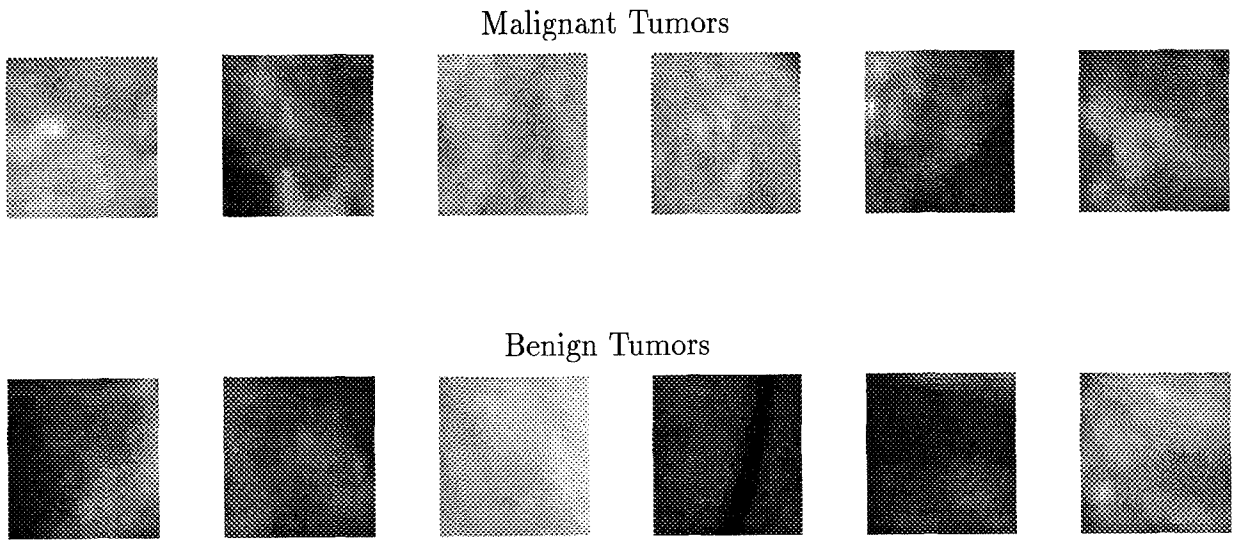


Figure 4.28 Examples of malignant and benign tumors.

Input Images	h	k
First pair	2	25
Second pair	5	20
Third pair	1	5
Fourth pair	1	10
Fifth pair	3	20
Sixth pair	1	20
All 6 pairs	5	35

Table 4.7 The h and k values for the malignant v. benign tumors.

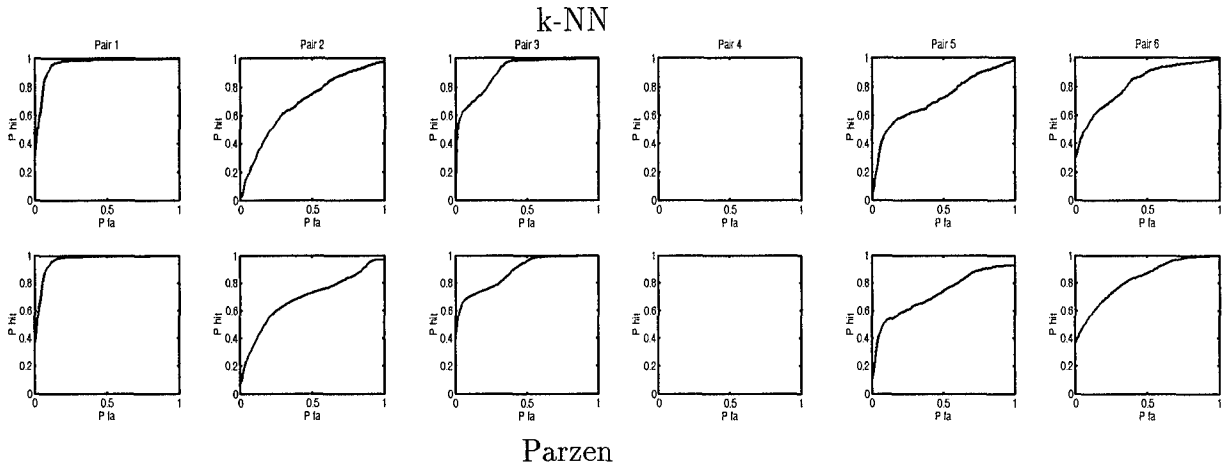


Figure 4.29 ROC curves for malignant versus benign tumors.

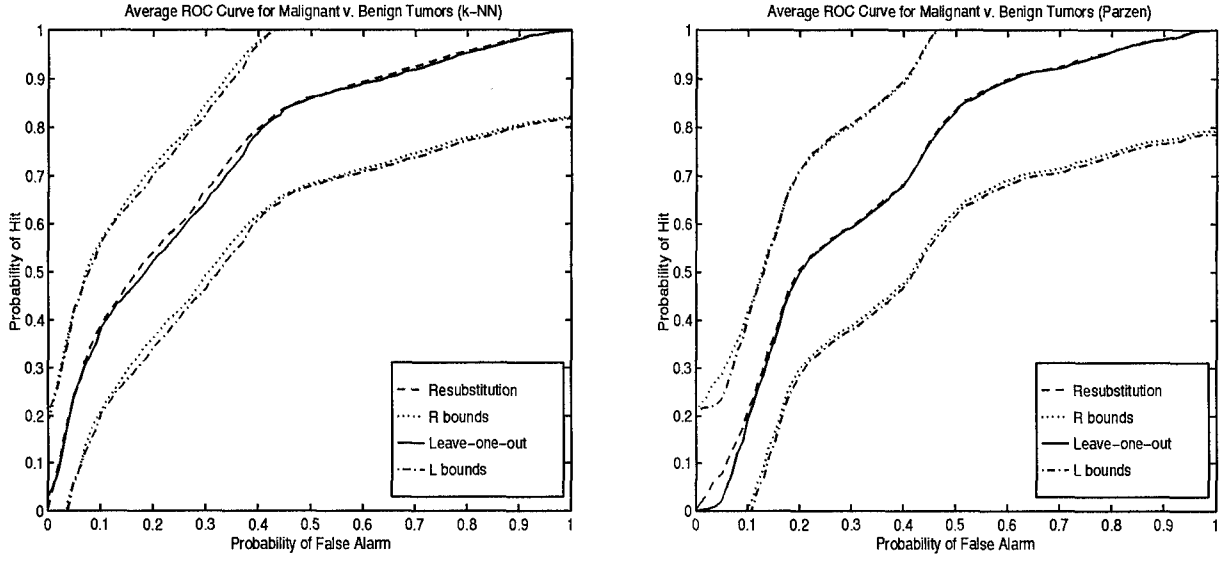


Figure 4.30 ROC curves for malignant versus benign tumors.

Input Images	ROC Area (k-NN)	ROC Area (Parzen)
First pair	.9569	.9629
Second pair	.6907	.6910
Third pair	.9019	.8842
Fourth pair	1	1
Fifth pair	.7157	.7234
Sixth pair	.8212	.8302
All 6 pairs	.7020	.7454

Table 4.8 Areas under the ROC for the malignant v. benign tumors.

curve for all six pairs combined is shown in Figure 4.32. The areas under the curves are listed in Table 4.10.

Input Images	h	k
First pair	1	30
Second pair	1	30
Third pair	3	35
Fourth pair	2	40
Fifth pair	2	50
Sixth pair	1	30
All 6 pairs	4	30

Table 4.9 The h and k values for the malignant v. benign tumors after angular 2^{nd} moment is performed.

When comparisons are made between the unprocessed images' ROC curves and areas and the angular 2^{nd} moment processed images' ROC curves and areas, it can be seen that the unprocessed images are better classified. It turns out that processing the images through the Khoros angular 2^{nd} moment spatial feature extraction did not improve the segmentation ability of the images. In fact, it made it worse.

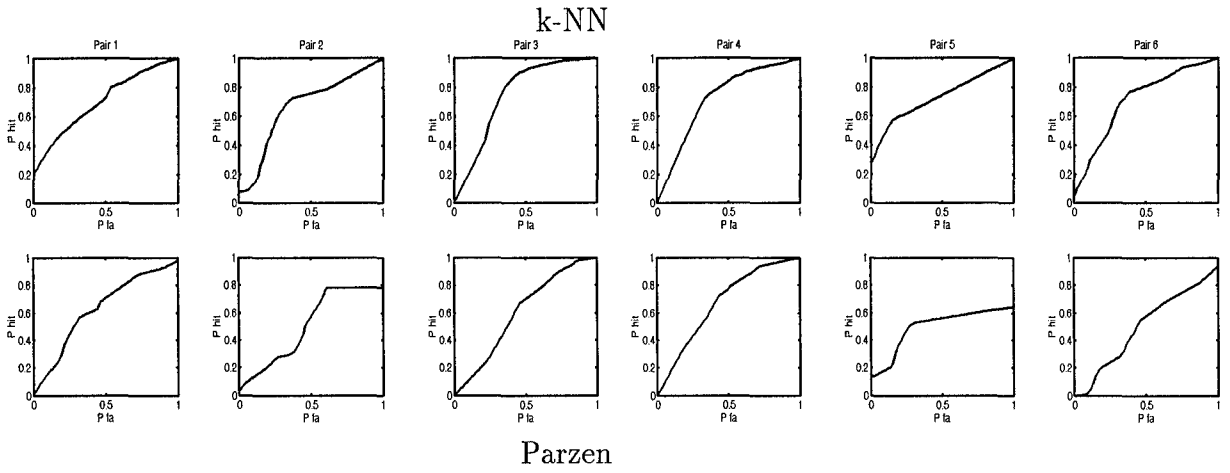


Figure 4.31 ROC curves for malignant versus benign tumors after angular 2^{nd} moment is performed.

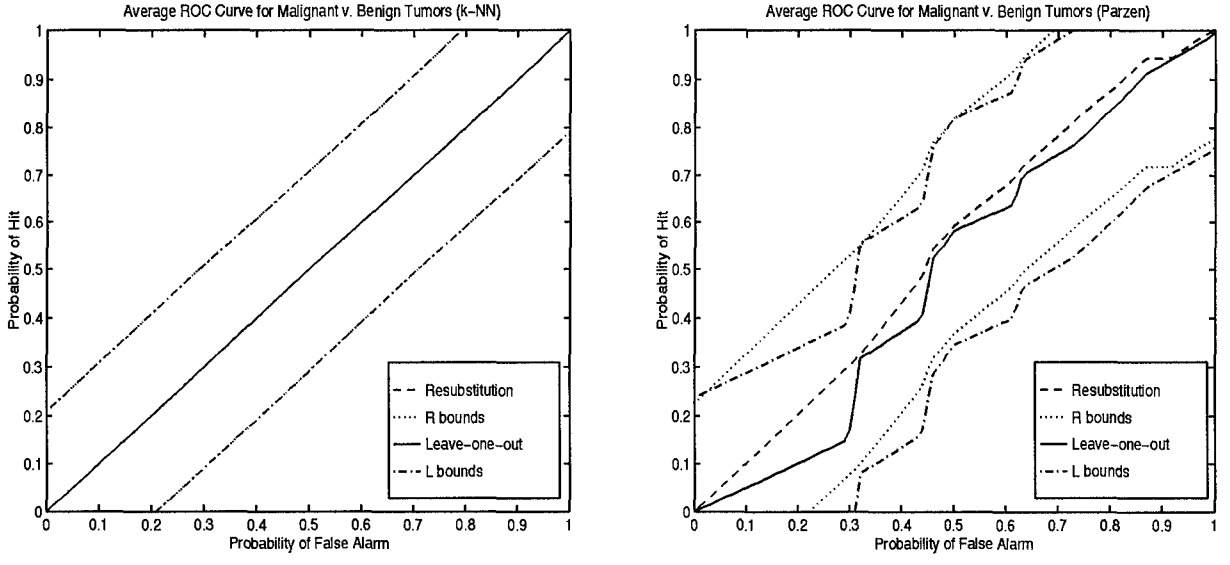


Figure 4.32 ROC curves for malignant versus benign tumors after angular 2^{nd} moment is performed.

Input Images	ROC Area (k-NN)	ROC Area (Parzen)
First pair	.7051	.6279
Second pair	.6691	.5071
Third pair	.7470	.6025
Fourth pair	.7226	.6701
Fifth pair	.7306	.4973
Sixth pair	.7113	.4987
All 6 pairs	.5000	.4969

Table 4.10 Areas under the ROC for the malignant v. benign tumors after angular 2^{nd} moment is performed.

4.7 Segmented Images

In this section each pixel in the image is classified as target or background, and a new, segmented image is created. The thresholds used to make the decisions are created by putting hand-segmented target and background matrices through the ROC code, choosing a desired hit probability, and finding the discriminant value that corresponds to that hit probability. After the threshold is chosen, discriminant values are calculated for each pixel. Each pixel is classified by determining which side of the discriminant threshold it falls on. A problem encountered was that the images were too large to compute all the new pixel values at once. So, each line in the image was done separately.

The original image is segmented using a hit probability of 80%. The morphological filtered image is segmented using a hit probability of 95%. These values are chosen from the ROC curves in Figures 4.6 and 4.15. The new segmented images can be seen in Figures 4.33 - 4.35. In the segmented images, the background pixels are black and the target pixels are white.

From the images, it can be seen that the morphological filtered image outperformed the original unprocessed image in segmentation. It is easy to see that

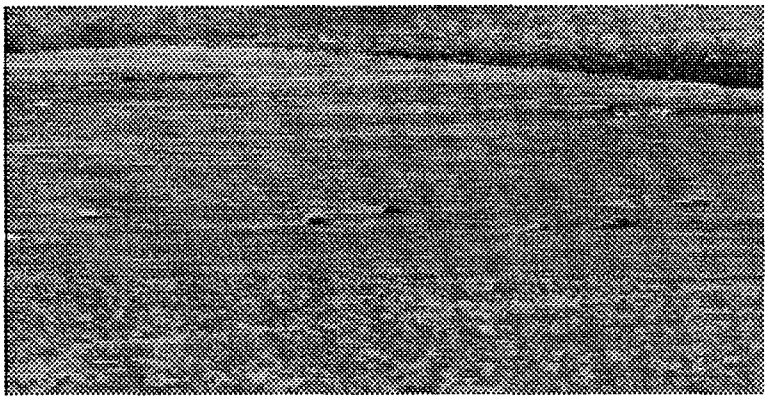


Figure 4.33 The original image.

Figure 4.34 The segmented original image.

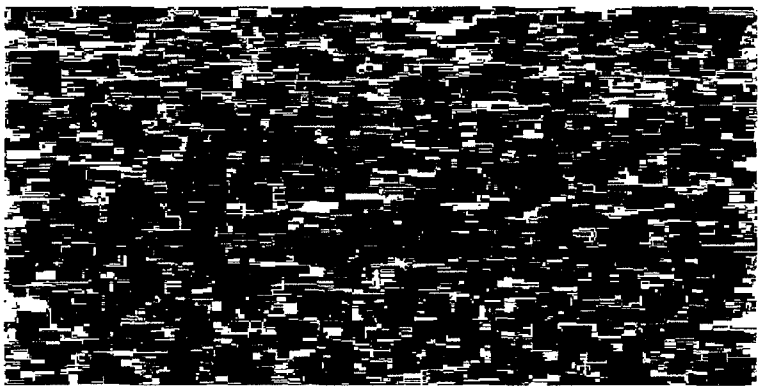


Figure 4.35 The segmented morphological filtered image.

more of the background is correctly identified as background. But even though this morphological filter helped in the segmentation, it still did not perform as well as had been hoped.

To take the segmentation process further, the segmented morphological filtered image is correlated with a 10x50 kernel and thresholded. These new segmented images, with two different threshold values (250 and 325), are shown in Figure 4.36. This extra processing has eliminated much of the background, but it still shows false alarms and misses.

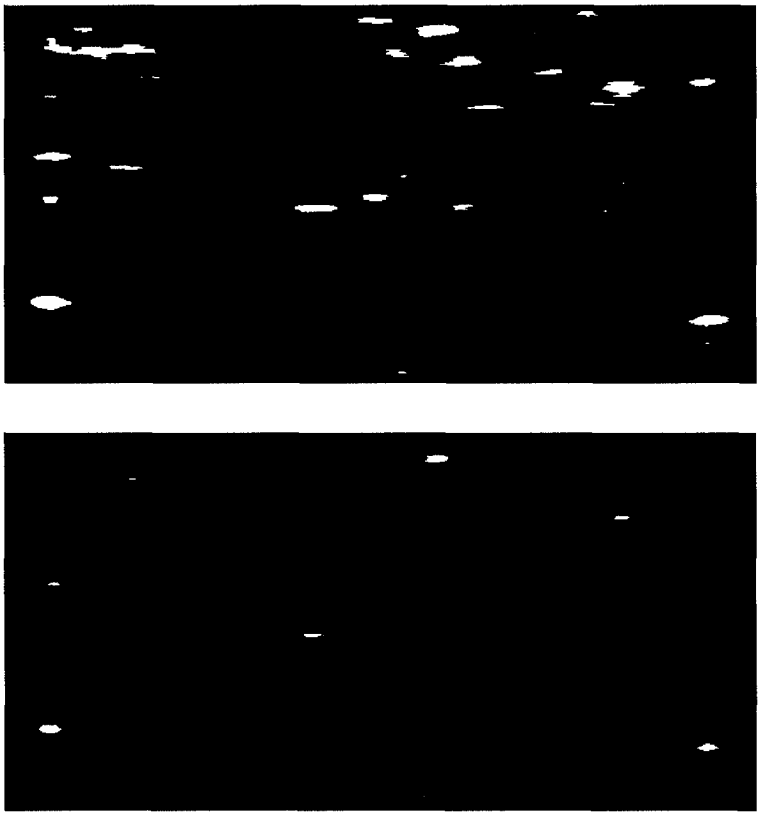


Figure 4.36 The correlated and thresholded morphological filtered image.

4.8 Summary

The tests performed using the ROC code showed that the estimates give a good approximation of the empirical ROC curves for the data. On that assumption, the algorithms were performed on real infrared images. These output ROC curves and the areas under them can be used as a measurement system to determine the best image processing algorithms for different images.

V. Conclusions

5.1 Introduction

The main objective of this research is to develop ROC curves to compare IR image processing techniques. The non-parametric techniques used to approximate the density functions and create the ROC curves were tested on problems with known distributions, then applied to the IR image problems. Chapter II provided the theory behind the methodology in Chapter III, and Chapter IV presented the resulting ROC curves. This chapter summarizes the results of this research.

5.2 Summary of Significant Results

5.2.1 Test Results. The test problems show that the ROC curves created from the density estimation process are good estimates of the empirical ROC curves of the given data sets.

5.2.2 IR Results. After showing that the estimated ROC curves approximate the true curves for the test data, the techniques are applied to actual IR imagery. In trying to identify target versus background pixels, the ROC curves indicate that combining the spatial features bands gives better results than using no processing on the image. Furthermore, the gray-scale CMO morphological operator provides better results than both the unprocessed image and the spatial feature bands. So in this case, the CMO is the better image processing technique.

Since the USAF is interested in finding scud missile launch vehicles, the ROC curve techniques are used to identify scud versus non-scud pixels in the same images. The results obtained were similar to those of the target versus background problem. The CMO algorithm is the better processing technique. One result to note is that the ROCs created for the target case are better than those for the scud case. This

is due to the fact that the other target pixels (tanks, trucks, etc.) are similar to the scud pixels, or at least more similar to the scud pixels than the background pixels.

5.2.3 Breast Cancer. The ROC estimation techniques are applied to the breast cancer x-ray images in order to show that it is possible to determine the hit and false alarm probabilities associated with finding breast cancer tumors.

This was taken one step further to the problem of determining the probabilities of malignant and benign tumors. The angular 2^{nd} moment was performed on the tumor images, and the ROCs obtained were compared with the ROCs from the unprocessed images. It turns out that calculating the angular 2^{nd} moment did not improve the segmentation ability of the tumors.

5.3 Conclusions

The results of this thesis lead to the following conclusions:

- The test data sets show that the estimated ROC curves developed in this research are a good approximation to the actual ROC curves from the images.
- The ROC curves can provide an effective means of measuring the performance of image processing and segmentation techniques.
- Another means of measuring the performance of image processing techniques is the area under the ROC curves. A larger area (with area=1 being the highest) generally signifies a better image processing or segmentation technique.
- The ROC analysis used here can be used on a variety of images in different applications, not just in the military target scenario.
- Of the image processing techniques used in this thesis, the CMO morphological filter produced the best results. For the target versus background problem, the areas under both the k-NN and Parzen estimated ROC curves were increased from .7770 and .7546 for the original image to .9744 and .9716 for the mor-

phological processed image. For the scud versus non-scud problem, these ROC areas were increased from .6015 and .5811 to .8528 and .8731.

5.4 Recommendations

This research can be continued in the following areas:

- Instead of calculating discriminant values to create a one-dimension decision problem, leave the data in n-dimensional space and use hyper-planes as the decision boundaries.
- Discover a way to find actual probabilities of hit and false alarm at the target level rather than the pixel level.

Appendix A. KHOROS

A.1 Cantata Workspace

The Cantata workspace is the image processing environment used in this report. To get this workspace, you must have the path set up. This should be done when you receive your account. If not, ask Dave Doak or Dan Zambon to help you set the path. Once this is done, just go into a command tool and type **cantata** to get to the cantata workspace shown in Figure A.1.

A.2 Input Images

In the workspace, there is a box called **input sources**. Pull down the menu from that box and you can either input an image file or create a file in Khoros. To input a file, choose **input data file**. When the input file window appears, you can choose any of the standard images, or you can use your own by clicking the **user defined** box and typing in the file name or using the browser. Use the browser by clicking on **user specified file**. This will take you to a file manager where you can choose your file. You can move through your directory and even get a file from a classmate's directory. Once you choose your file, click on the **glyph** box. This will create an input glyph containing your input image that you can position on the workspace with your mouse.

A.3 Using Glyphs

Once you have an input image, you can begin processing it by selecting other glyphs and connecting them. Connect them by clicking the right side arrow on one glyph to the left side arrow of the next glyph. An example of connected glyphs in a workspace is shown in Figure A.2. You can run your program by clicking on the **run** box or by turning on each glyph separately using its on/off switch.

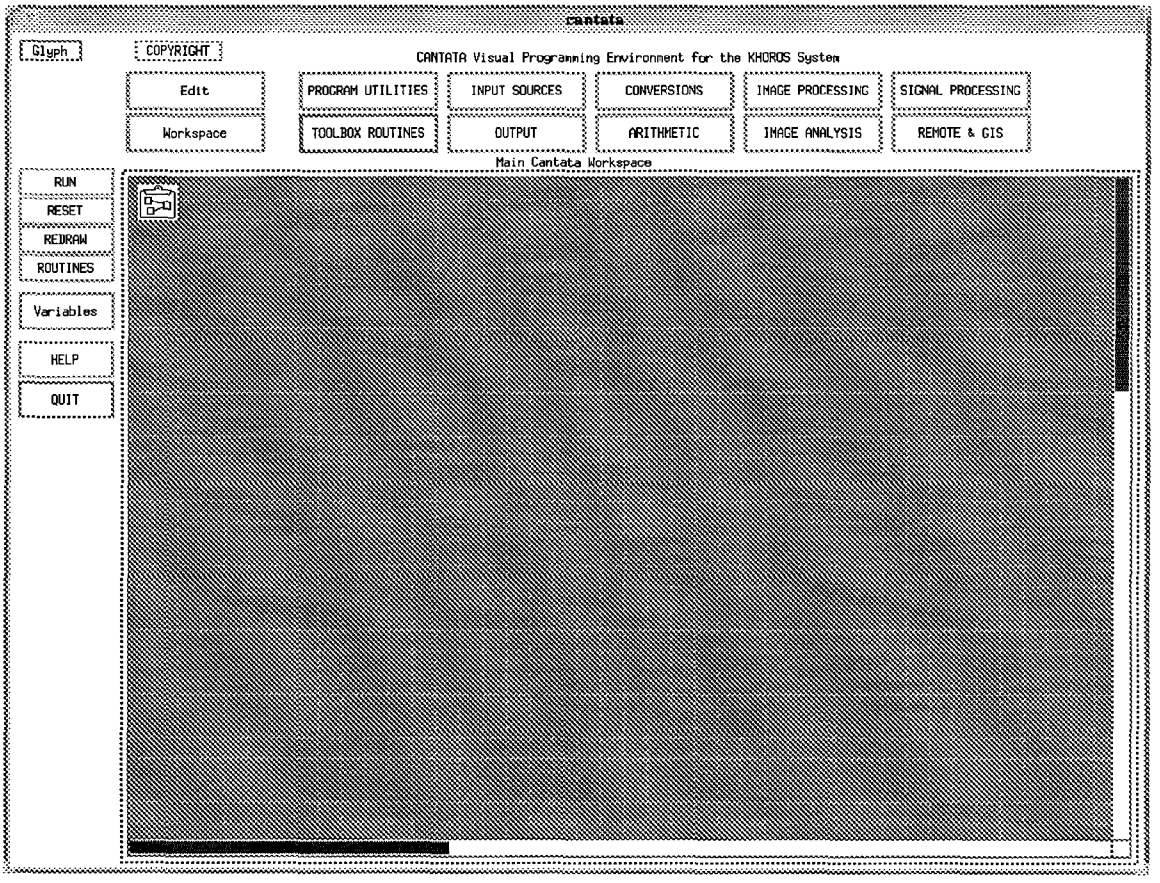


Figure A.1 The Cantata workspace.

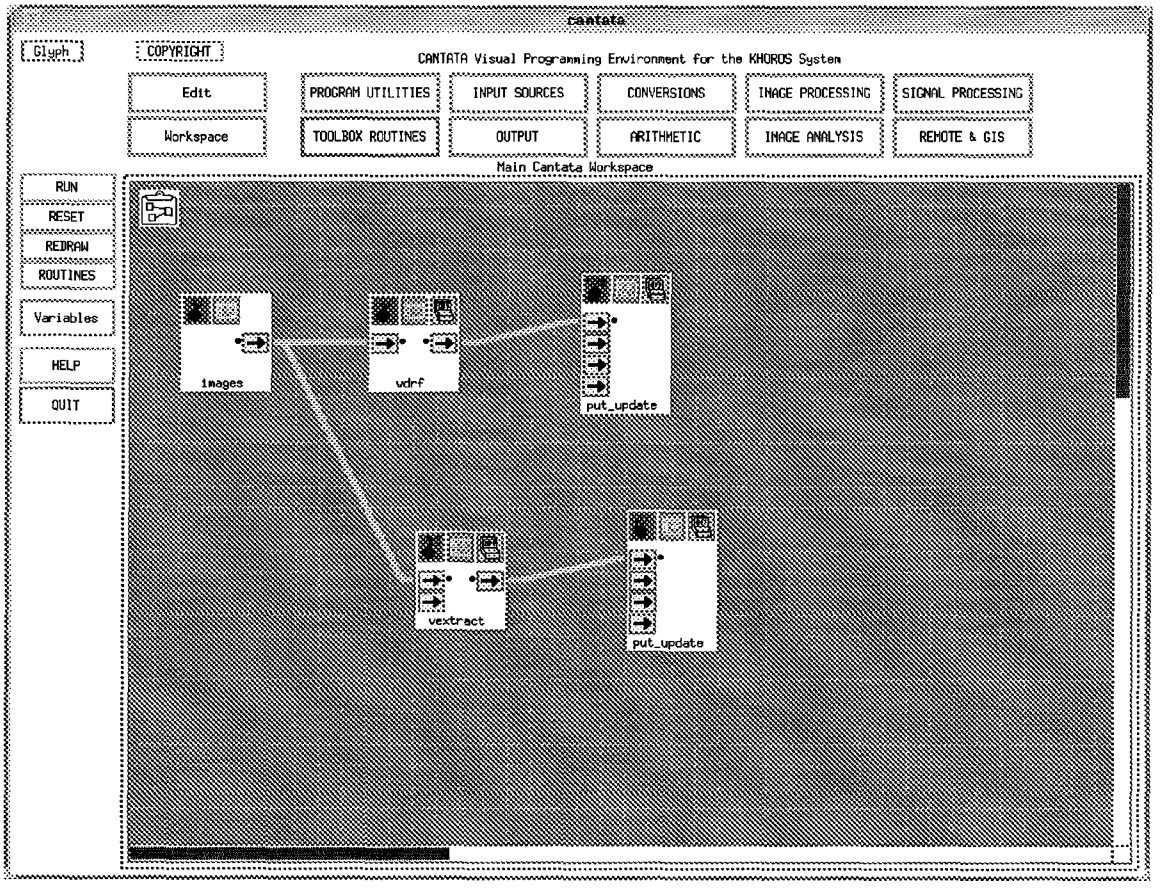


Figure A.2 Cantata workspace glyphs.

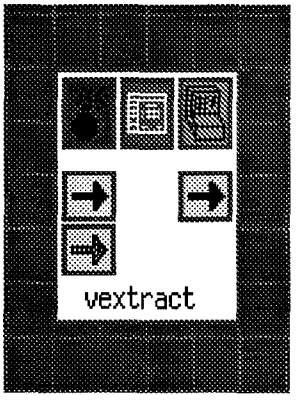


Figure A.3 An example glyph.

An example glyph can be seen in Figure A.3. Use the arrows to connect the glyph to other glyphs in the workspace. All of the dark arrows must be connected to something or there will be an error, but the light arrows are only used in some options for the glyph. To delete a glyph, click on the bomb symbol in the upper left corner. To turn a glyph on, click on the switch in the upper right corner. To change any of the settings on the glyph, click on the top middle box to get the menu window for that glyph.

A.4 Commonly Used Functions

This section contains information on where to find some useful commands, as well as some examples and a few things you should know about them. This is only a partial list, but it contains the ones I found most useful.

A.4.1 Convolution (2D). You can find convolution under **image processing, spatial filters**. You just need to input an image and a kernel. A faster way to convolve if you have large images is to take the FFT, multiply, and inverse FFT. To show you what to expect as output, the two images in Figure A.4 have been convolved to give the output seen in Figure A.5. The dimensions of the images shown here are proportional to the actual dimensions of the images that were used.

A.4.2 Correlation (2D). There is not a built in correlation function. To perform a correlation, flip the input kernel image (both ways using the flip option) and perform a convolution. If you have complex images, you must conjugate and flip the kernel. An example of the correlation of the two images in Figure A.4 is shown in Figure A.6.

A.4.3 Data Conversion. Sometimes, you need to convert data from one type to another. Usually, Khoros will let you know by giving you an error and

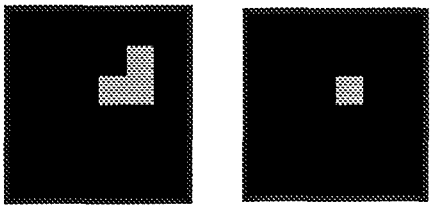


Figure A.4 The input image and kernel.

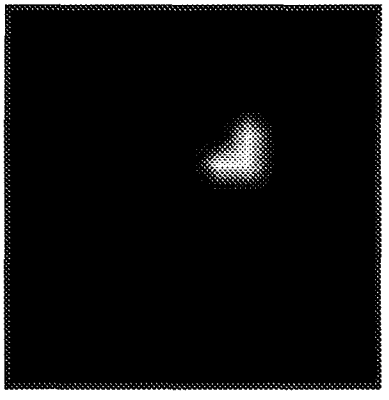


Figure A.5 The resulting convolution.

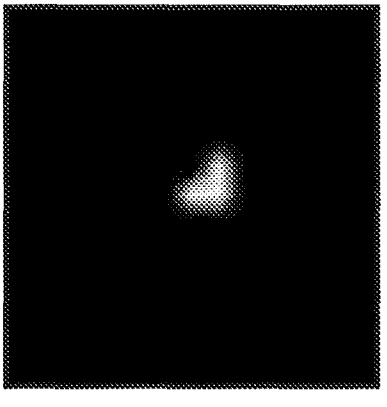


Figure A.6 The resulting correlation.

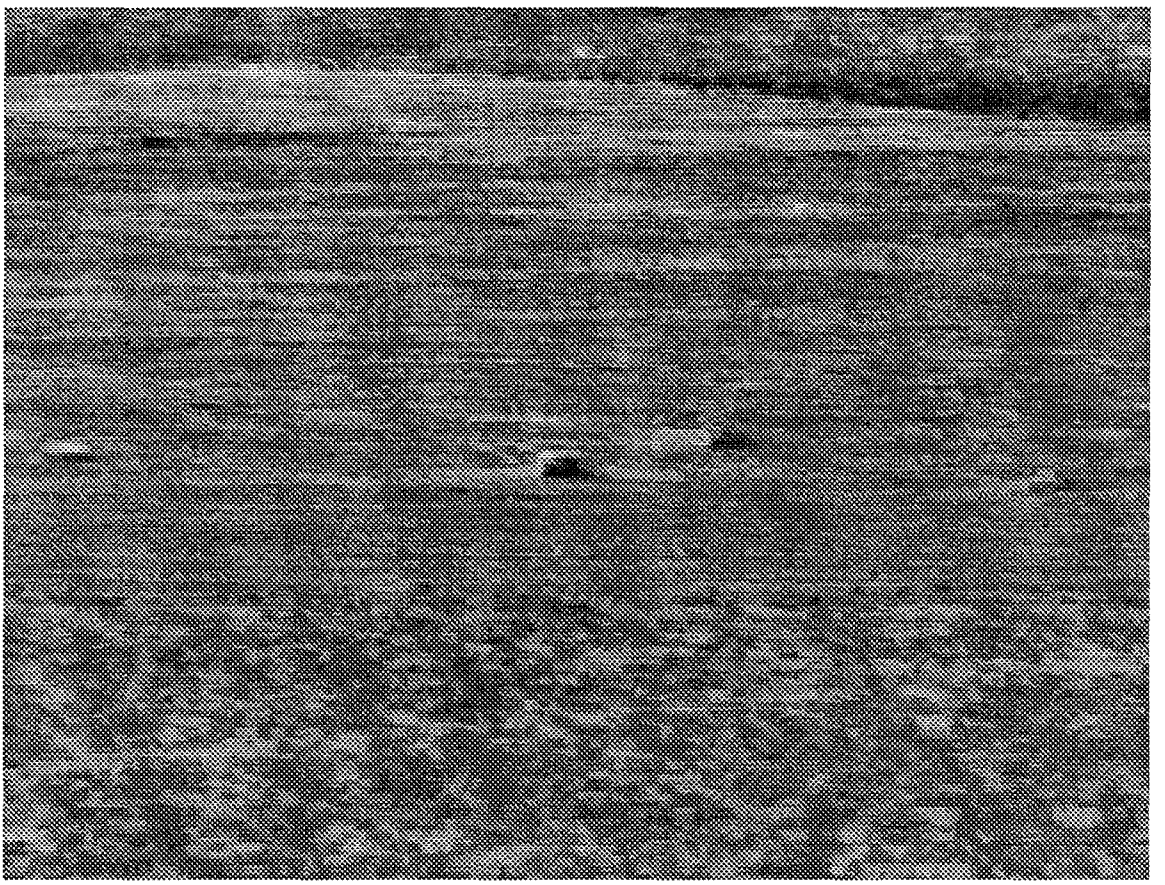


Figure A.7 An example image.

telling you which type it needs. Then just go to **conversions, data conversion** and choose the type you need.

A.4.4 Expand. To enlarge an image, click on **image processing, geometric manipulations, expand image**. Then choose a scale factor. Be aware that resolution does not increase. Each pixel just gets bigger, as you can see in Figure A.8.

A.4.5 Extract Part of an Image. Look under **image processing, subregion**. You can determine the coordinate values you desire by running the mouse over the original image. An example of a segmented image is shown in Figure A.8.

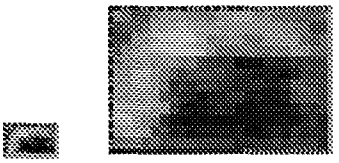


Figure A.8 An extracted image and corresponding expanded image.

A.4.6 File Formats. If you want to input or export files from between different programs such as Khoros and Matlab, just do a file conversion by clicking on **conversions, standard file format** and then choose the file type you need.

If you want to convert between raw or ascii and VIFF files, look under **conversions, raw file format**.

A.4.7 Filters (Frequency). Frequency filters are found under **image processing, frequency filters**. These assume that your input image is already in the frequency domain. An example output of a high-pass frequency filter is shown in Figure A.10.

A.4.8 Filters (Spatial). Spatial filters are found under **image processing, spatial filters** These assume that the input image is in the space domain. The edge extraction filters have the same effect as converting to the frequency domain and doing high-pass frequency filtering. An example of this is shown in Figure A.10.

A.4.9 Flip. Look under **image processing, geometric manipulations**. Input the image and choose which way you want to flip it. For a correlation, flip the image both top-for-bottom and right-for-left.

A.4.10 Fourier Transform (FFT & IFFT). The 2D Fourier transform is located under **image processing, transforms**. You can choose either forward or

Figure A.9 Lenna.

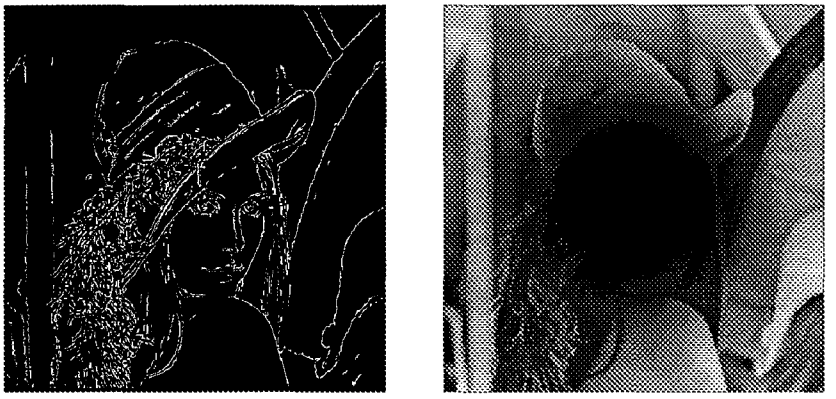


Figure A.10 Lenna after using an edge-extract spatial filter and high-pass frequency filter.

inverse. One requirement is that the input images have dimensions that are a power of two, so you may need to pad them.

A.4.11 Multiplication. Look under **arithmetic, binary arithmetic**. This performs a pixel by pixel multiplication of the images. When you multiply two images, they must be the same size.

A.4.12 Pad an Image. Look under **image processing, subregion**. Input the new row size and column size of the padded image.

A.4.13 Spatial Feature Band Extraction. The Khoros image processing environment has a built-in function to extract spatial features from images. These features are the mean, variance, contrast, angular 2^{nd} moment, entropy, and dispersion. The following equations describe how each spatial feature is calculated. The algorithm assumes there are K gray levels in the image and $p(k)$ = the probability of occurrence of each gray level k.

$$\text{mean}, \mu = E[k] = \sum_{k=0}^{K} k\, p(k) \tag{A.1}$$

$$\text{variance} = E[k^2] - E[k]^2 = \sum_{k=0}^{K} k^2\, p(k) - \left[\sum_{k=0}^{K} k\, p(k)\right]^2 \tag{A.2}$$

$$\text{contrast} = \sum_{k=0}^{K} k^2\, p(k) \tag{A.3}$$

$$\text{angular } 2^{\text{nd}} \text{ moment} = \sum_{k=0}^{K} p^2(k) \tag{A.4}$$

$$\text{entropy} = -\sum_{k=0}^{K} p(k) \log_2 p(k) \tag{A.5}$$

$$\text{dispersion} = \sum_{k=0}^{K} |k - \mu|\ p(k) \tag{A.6}$$

Figure A.11 shows how the Khoros glyphs are connected in order to extract the spatial feature bands from the image. The **input** glyph loads the image, **arf2viff** converts the input data to the VIFF type used in Khoros, **vextract** is used to extract the 356x700 image, and **vconvert** converts the data type to byte which is needed in **vspatial**. The menu in the **vspatial** glyph can be used to select the window size and the spatial feature bands to be extracted. The **vbandspt3** glyphs extract each spatial band separately so it can be stored as a separate file through the **viff2mat** glyph operation. After removing the square brackets at the beginning and ending of the files, they can be loaded into Matlab. These new image band files are used in the classifying code.

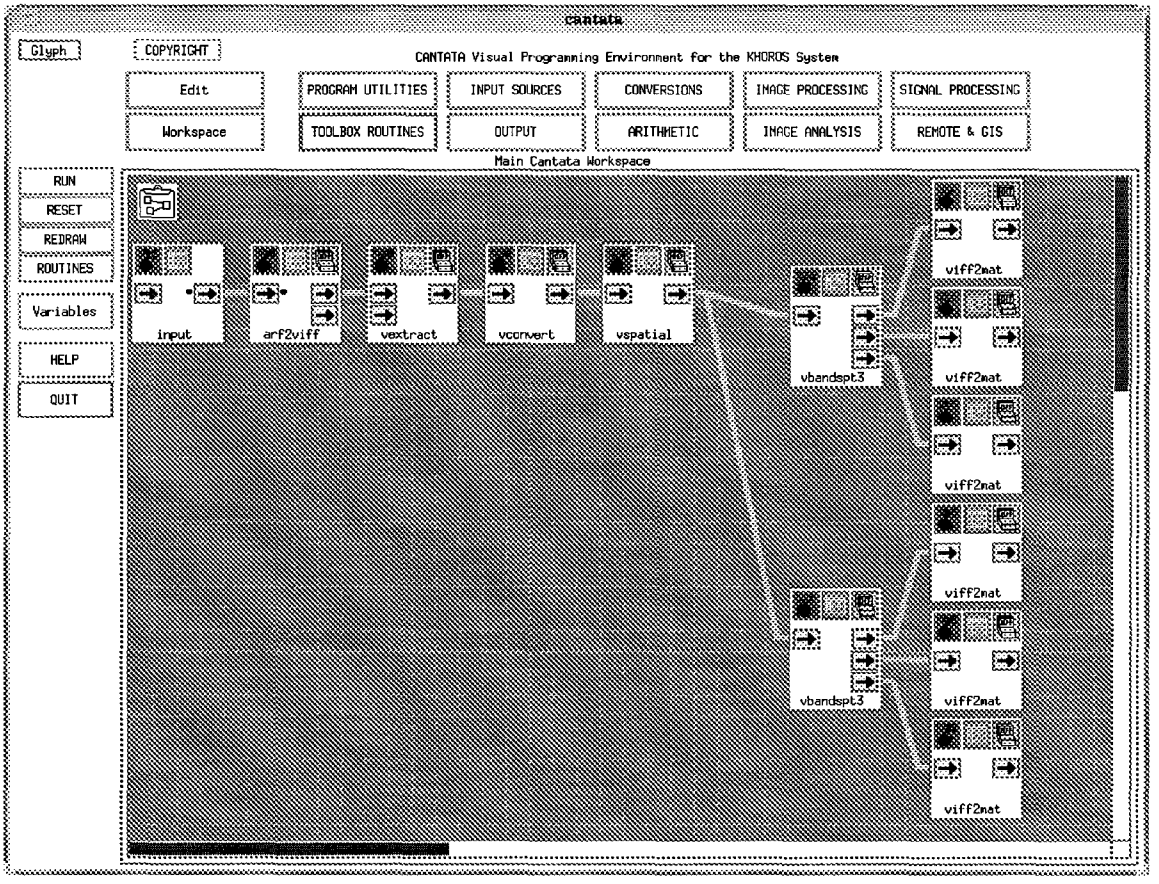


Figure A.11 A Cantata workspace showing the Khoros glyphs used to extract spatial features.

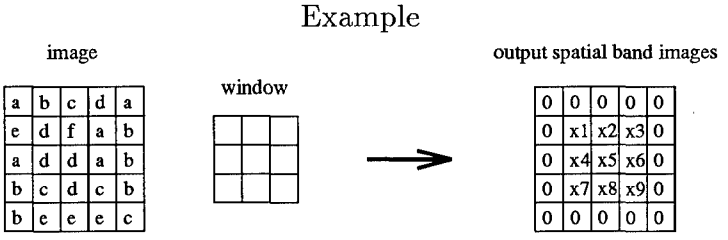


Figure A.12 How the Khoros spatial bands are created.

Using the example in Figure A.12 where a through f are various gray-scale values, the following equations show the calculations for the first shift of the window (where it covers the nine pixels in the upper left corner.)

mean: $x1 = \frac{a+b+c+e+d+f+a+d+d}{9}$

variance: $x1 = \frac{a^2+b^2+c^2+e^2+d^2+f^2+a^2+d^2+d^2}{9} - \left(\frac{a+b+c+e+d+f+a+d+d}{9}\right)^2$

contrast: $x1 = a^2\left(\frac{2}{9}\right) + b^2\left(\frac{1}{9}\right) + c^2\left(\frac{1}{9}\right) + d^2\left(\frac{3}{9}\right) + e^2\left(\frac{1}{9}\right) + f^2\left(\frac{1}{9}\right)$

ang 2^{nd} moment: $x1 = \left(\frac{2}{9}\right)^2 + \left(\frac{1}{9}\right)^2 + \left(\frac{1}{9}\right)^2 + \left(\frac{3}{9}\right)^2 + \left(\frac{1}{9}\right)^2 + \left(\frac{1}{9}\right)^2$

entropy: $x1 = -\left(\frac{2}{9}\right)\log_2\left(\frac{2}{9}\right) - \left(\frac{1}{9}\right)\log_2\left(\frac{1}{9}\right) - \left(\frac{1}{9}\right)\log_2\left(\frac{1}{9}\right) - \left(\frac{3}{9}\right)\log_2\left(\frac{3}{9}\right) - \left(\frac{1}{9}\right)\log_2\left(\frac{1}{9}\right) - \left(\frac{1}{9}\right)\log_2\left(\frac{1}{9}\right)$

dispersion: $x1 = |a-\mu|\left(\frac{2}{9}\right) + |b-\mu|\left(\frac{1}{9}\right) + |c-\mu|\left(\frac{1}{9}\right) + |d-\mu|\left(\frac{3}{9}\right) + |e-\mu|\left(\frac{1}{9}\right) + |f-\mu|\left(\frac{1}{9}\right)$, where $\mu = \frac{a+b+c+e+d+f+a+d+d}{9}$

A.5 User Defined Functions

In the image processing I have been doing, it was necessary to bring in a program that could convert ARF image files to VIFF files. The program was obtained through Wright Lab and was already put into a form suitable for bringing it into Khoros. I'm not sure exactly how it was done. All I had to do was to click on **workspace** and then **file utilities** to get a menu. Then I clicked on **UIS file** and got the browser where I chose the file I wanted to bring into Khoros. Doing this created a glyph that could be manipulated like any other glyph in Khoros.

A.6 Printing an Image

To print an image directly from Khoros, you can use the menu to choose **output** and then **print image.** Choose the format you need and then in the **send to printer** box, type | **lpr -Psparc3** (or any printer of your choosing.) Then run that glyph.

An alternative to this is to save the image into your directory as a .ps file then you can use a regular print tool to print the image. To do this, select from the Khoros menu **conversions** then **standard file format.** When you get the new window, select **VIFF to Raster**, the Khoros input file, and the name for the output file. This will create a raster (.rs) file in your directory. Then just convert this to a .ps file. This can be done in two ways:

1. In a command tool, type **convert oldfilename.rs newfilename.ps.** The new postscript file will appear in your file manager.
2. In a command tool, type **xv filename.rs.** The image will appear. To get a menu of options, click the left mouse button on the image. When saving the image, you can choose the output file type postscript.

A.7 Workspace Utilities

A.7.1 Save a Workspace. To save a workspace, pull down the **workspace** menu in cantata. Choose **file utilities** and enter an output file name by typing it in (or using the browser if the file already exists). After entering the file name, press return.

A.7.2 Restore a Workspace. To restore a saved workspace, again choose **workspace** followed by **file utilities.** Then enter an input file name by typing it in or using the browser. The saved workspace will be restored to the screen.

Appendix B. Morphology Code

B.1 driver script file

```
% This script file performs the close minus open (CMO) algorithm on an image.
% (Uses image A and morphological kernel B to produce the CMO)

load origimage.dat;

A=origimage;
B=ones(10,50);

c=closing(A,B);
o=opening(A,B);
imageout=c-o;
save morph10x50 imageout
```

B.2 closing.m

```
% CLOSING: This function performs the gray-scale closing of an image by a
%          structuring element
%
% Inputs:  A : image
%          B : structuring element
%
% Output:  c : closed image
%

function[c]=closing(A,B)
temp=dilation(A,B);
c=erosion(temp,B);
```

B.3 opening.m

```
% OPENING:  This function finds the opening of a gray-scale image A by the
%           structuring element B.
%
% Inputs:  A : image
%          B : structuring element
%
% Output:  c : opened image
%

function[c]=opening(A,B)
temp=erosion(A,B);
c=dilation(temp,B);
```

B.4 dilation.m

```
% DILATION: This function finds the gray-scale dilation of an image by a
%           structuring element.
%
%           [dilatedimage] = dilation(A,B)
%
% Inputs:  A : image
%          B : structuring element
%
% Output:  dil: dilated image
%

function[dil]=dilation(A,B)
[Arow,Acol]=size(A);
[Brow,Bcol]=size(B);

newA=zeros(Arow+2*Brow-2,Acol+2*Bcol-2);
newA(Brow:Arow+Brow-1,Bcol:Acol+Bcol-1)=A;

for s=Brow:Arow+2*Brow-2
    for t=Bcol:Acol+2*Bcol-2
        dil(s-Brow+1,t-Bcol+1)=max(max(newA(s:-1:s-Brow+1,t:-1:t-Bcol+1)+B));
    end
end
```

B.5 erosion.m

```
% EROSION: This function finds the gray-scale erosion of an image by a
%          structuring element.
%
% Inputs:  A : image
%          B : structuring element
%
% Output:  ero: eroded image
%

function[ero]=erosion(A,B)
[Arow,Acol]=size(A);
[Brow,Bcol]=size(B);

for s=1:Arow-Brow+1
    for t=1:Acol-Bcol+1
        ero(s,t)=min(min(A(s:s+Brow-1,t:t+Bcol-1)-B));
    end
end
```

Appendix C. Martin's Code

C.1 driver script file

```
% Driver script file used to run Martin's code and create the plots

load targetset;
load backgroundset;

k=2:50;
h=.1:.1:10;

opt=1;
[Rp1, Lp1, Rk1, Lk1] = pknn(targetset,backgroundset, h, k, opt);

opt=2;
[Rp2, Lp2, Rk2, Lk2] = pknn(targetset,backgroundset, h, k, opt);

mRp1=mean(Rp1);
mLp1=mean(Lp1);
mRk1=mean(Rk1);
mLk1=mean(Lk1);
[minp1,ip1] = min(mLp1);
[mink1,ik1] = min(mLk1);

mRp2=mean(Rp2);
mLp2=mean(Lp2);
mRk2=mean(Rk2);
mLk2=mean(Lk2);
[minp2,ip2] = min(mLp2);
[mink2,ik2] = min(mLk2);

figure(1)
subplot(2,1,1)
plot(k, mLk1, '--', k, mRk1, '-'); hold on
plot([k(ik1) k(ik1)], [mLk1(ik1) mRk1(ik1)], 'w*');
plot(k, mLk2, ':');
plot([k(ik2) k(ik2)], [mLk2(ik2) mRk2(ik2)], 'w*');
legend('-','Resubstitution','--','Leave-one-out (opt1)',':','Leave-one-out (opt2)');
ylabel('Average Error (%)'); xlabel('k');
title('k-NN'); hold off

subplot(2,1,2)
plot(h, mLp1, '--', h, mRp1, '-'); hold on
plot([h(ip1) h(ip1)], [mLp1(ip1) mRp1(ip1)], 'w*');
ylabel('Average Error (%)'); xlabel('h');
title('Parzen');
legend('-','Resubstitution','--','Leave-one-out (opt1)',':','Leave-one-out (opt2)');
plot(h, mLp2, ':');
plot([h(ip2) h(ip2)], [mLp2(ip2) mRp2(ip2)], 'w*'); hold off
```

C.2 *pknn.m*

```
% PKNN:  Run Parzen and kNN procedure for X1, X2 10 times.
%
%     [Rp, Lp, Rk, Lk] = PKNN(X1, X2, h, k, opt)
%
% Inputs:  X1, X2:  data sets (n x N1 and n x N2)
%          h, k:    values to use for h and k
%          opt:     1 = threshold option 3
%                   2 = threshold option 4
%
%
% All h's must be greater than zero, and k must be between 2 and
% min(N1, N2)-1
%
% Outputs: Rp, Lp: Parzen R and L errors for each test (one test per row)
%          Rk, Lk: k-NN R and L errors for each test

function [Rp, Lp, Rk, Lk] = pknn(X1, X2, h, k, opt)

[n1, N1] = size(X1);
[n2, N2] = size(X2);
if n1 ~= n2
    fprintf(2, 'Data sets X1 and X2 must have same number of rows
(features)\n');
    return;
end  % if

% Keep this value on hand
dim = n1;
ntests = 10;

Rp = zeros(ntests, size(h,2));
Lp = zeros(ntests, size(h,2));
Rk = zeros(ntests, size(k,2));
Lk = zeros(ntests, size(k,2));

fprintf(1, 'Threshold Option = %d \n', opt);
fprintf(1, 'Performing %d independent tests: \n', ntests);

for test = 1:ntests,

    fprintf(1,' Test #%d:\n', test);

    testsamps = test:ntests:N1;
    t1 = [1, testsamps+1];
    t2 = [testsamps-1, N1];
    others = [];
    for i = 1:size(t1,2),
        others = [others, t1(i):t2(i)];
    end  % for i

    x1 = X1(:, testsamps);
    iS1 = cov(X1(:,others)');

    testsamps = test:ntests:N2;
    t1 = [1, testsamps+1];
    t2 = [testsamps-1, N2];
    others = [];
    for i = 1:size(t1,2),
        others = [others, t1(i):t2(i)];
    end  % for i

    x2 = X2(:, testsamps);
    iS2 = cov(X2(:,others)');
```

```
clear testsamps others t1 t2

n1 = size(x1, 2);
n2 = size(x2, 2);
fprintf(1, '  %d Class 1 samples, %d Class 2 samples\n', n1, n2);

fprintf(1, '  Estimating and inverting covariance matrices ...\n');
detratio = -0.5 * log(det(iS2)/det(iS1));
iS1 = inv(iS1);
iS2 = inv(iS2);

fprintf(1, '  Computing distances ...\n');

[d11, d12, d21, d22] = compute_distances(x1, x2, iS1, iS2);

fprintf(1, '  Classifying (Parzen) ...\n');
Rerr = [];
Lerr = [];

for r = h,

    % Compute sums:
    temp = -0.5 / r^2;
    s11 = sum(exp(temp * d11));
    s12 = sum(exp(temp * d12));
    s21 = sum(exp(temp * d21));
    s22 = sum(exp(temp * d22));

    lr1 = detratio - log((n2 * s11) ./ (n1 * s21));
    lr2 = detratio - log((n2 * s12) ./ (n1 * s22));

    ll1 = detratio - log((n2 * (s11 - 1)) ./ ((n1 - 1) * s21));
    ll2 = detratio - log(((n2 - 1) * s12) ./ (n1 * (s22 - 1)));

    [rerr, lerr] = classify(lr1, lr2, ll1, ll2, opt);

    Rerr = [Rerr, rerr];
    Lerr = [Lerr, lerr];

end  % for r

Rp(test,:) = 100 * Rerr / (n1+n2);
Lp(test,:) = 100 * Lerr / (n1+n2);


fprintf(1,'  Classifying (k-NN) ...\n');

% sort distances
d11 = sort(d11);
d12 = sort(d12);
d21 = sort(d21);
d22 = sort(d22);

tr = detratio + log(n1/n2);
tl1 = detratio + log((n1-1)/n2);
tl2 = detratio + log(n1/(n2-1));
Rerr = [];
Lerr = [];

for i = k,

    lr1 = tr + 0.5 * dim * log(d11(i,:) ./ d21(i,:));
    lr2 = tr + 0.5 * dim * log(d12(i,:) ./ d22(i,:));
```

```
        ll1 = tl1 + 0.5 * dim * log(d11(i+1,:) ./ d21(i,:));
        ll2 = tl2 + 0.5 * dim * log(d12(i,:) ./ d22(i+1,:));

        [rerr, lerr] = classify(lr1, lr2, ll1, ll2, opt);

        Rerr = [Rerr, rerr];
        Lerr = [Lerr, lerr];

    end  % for i

    Rk(test,:) = 100 * Rerr / (n1+n2);
    Lk(test,:) = 100 * Lerr / (n1+n2);

end  % for test
```

C.3 compute_distances.m

```
% COMPUTE_DISTANCES:  Compute K distances between samples in X1 and X2.
%
%     [D11, D12, D21, D22] = COMPUTE_DISTANCES(X1, X2, invS1, invS2)
%
%   D11 = class 1 distances for samples of class 1
%   D12 = class 2 distances for samples of class 1
%   D21 = class 1 distances for samples of class 2
%   D22 = class 2 distances for samples of class 2
%
% invS1 and invS2 are the covariance matrix inverses (maybe estimated).
% X1 and X2 have one observation per column.

function [D11, D12, D21, D22] = compute_distances(X1, X2, invS1, invS2)

% Number of columns is the number of samples
N1 = size(X1,2);
N2 = size(X2,2);

% Allocate space
D11 = zeros(N1,N1);
D12 = zeros(N1,N2);
D21 = zeros(N2,N1);
D22 = zeros(N2,N2);

% Calculate intra-class distances:
  % Class 1:
for i = 1:N1-1
    for j = i+1:N1
D11(i,j) = (X1(:,j) - X1(:,i))' * invS1 * (X1(:,j) - X1(:,i));
D11(j,i) = D11(i,j);
    end
end

  % Class 2:
for i = 1:N2-1
    for j = i+1:N2
D22(i,j) = (X2(:,j) - X2(:,i))' * invS2 * (X2(:,j) - X2(:,i));
D22(j,i) = D22(i,j);
    end
end

% Calculate inter-class distances:
for i = 1:N1              % Rows for D12, Columns for D21
    for j = 1:N2          % Columns for D12, Rows for D21
% Pull samples out of matrices only once
v = X2(:,j) - X1(:,i);
D12(i,j) = v' * invS1 * v;

% These could be (-v), but there's no reason for it. (note (j,i))
D21(j,i) = v' * invS2 * v;
    end
end
```

C.4 classify.m

```
% CLASSIFY: Select thresholds and classify resubstitution and
%           leave-one-out discriminant values.
%
%     [R, L] = CLASSIFY(Lr1, Lr2, Ll1, Ll2, option)
%
% Inputs:  Lr1, Lr2: resubstitution discriminant values
%          Ll1, Ll2: leave-one-out discriminant values
%          option:   1 = threshold option 3
%                    2 = threshold option 4
%
% Outputs: R: Resubstitution error
%          L: Leave-one-out error

function [R, L] = classify(Lr1, Lr2, Ll1, Ll2, option)

% First get the minimum resubstitution error and threshold
fprintf(1,' Resubstitution...');
[R, t] = min_error(Lr1, Lr2);
fprintf(1,'done.\n');

% Now, depending on the option, get the minimum leave-one-out error
fprintf(1,' Leave one out...');
if option == 1
    L = sum(Ll1 > t) + sum(Ll2 < t);
else
    n1 = size(Ll1, 2);
    n2 = size(Ll2, 2);
    L = 0;
fprintf(1,'Class 1...');
    for i = 1:n1,
        [err, t] = min_error(Ll1([1:i-1 i+1:n1]), Ll2);
        if Ll1(i) > t
            L = L + 1;
        end  % if Ll1(i) > t
    end  % for i
fprintf(1,'Class 2...');
    for i = 1:n2,
        [err, t] = min_error(Ll1, Ll2([1:i-1 i+1:n1]));
        if Ll2(i) < t
            L = L + 1;
        end  % if Ll2(i) < t
    end  % for i
end  % if option == 1
fprintf(1,'done.\n');
```

C.5 *min_error.m*

```
% MIN_ERROR: Select threshold which gives the minimum error for
%            classifying two sets of discriminants.
%            This is designed to be used for any number of elements
%            in each set.  In the case of several minimum errors,
%            the threshold that yields the same number of
%            misclassifications from set 1 and set 2 is selected.
%
%     [ERROR, THRESHOLD] = MIN_ERROR(SET1, SET2)
%
% Inputs:  SET1, SET2: sets of discriminants from classes 1 & 2
%
% Outputs: ERROR: Minimum error obtainable
%          THRESHOLD: "Best?" threshold yielding the minimum error

function [error, threshold] = min_error(set1, set2)

big = realmax - realmin;
search = 0;

% Eliminate infinities from the search.
infs = find(set1==inf);
set1(infs) = big * ones(size(infs));
infs = find(set1==(-inf));
set1(infs) = -big * ones(size(infs));
infs = find(set2==inf);
set2(infs) = big * ones(size(infs));
infs = find(set1==(-inf));
set2(infs) = -big * ones(size(infs));

a = min(set1);
b = max(set1);
c = min(set2);
d = max(set2);

if b < c
    threshold = 0.5 * b + 0.5 * c;
    error = 0;
    return;
end  % if

if (min([a b c d]) == c) & (max([a b c d]) == b)
    %pick c or b at random.
    choice = round(rand);
    if choice == 0
        threshold = b + realmin;
    else
        threshold = c - realmin;
    end  % if choice
    error = sum(set1 > threshold) + sum(set2 < threshold);
    fprintf(1,'Bad decision rule: a=%d, b=%d, c=%d, d=%d\n',a,b,c,d);
    return;
end  % if

if (a < c & c < d & d < b) | ( c < a & a < b & b < d )
    lo = a;
    hi = d;
elseif (a < c & c < b & b < d)
    lo = c;
    hi = b;
else
    % What's left??? Write a message.write out a, b, c, and d
    fprintf(1,'Equal condition: a=%d, b=%d, c=%d, d=%d\n',a,b,c,d);
    threshold = .25 * a + .25 * b + .25 * c + .25 * d;
```

```
    error = -1;
    return;
end  % if

% go fish (in a limited pool)
pool = [set1(set1>=lo & set1<=hi) set2(set2>=lo & set2<=hi)];

% Get one copy of each distinct value in temp
temp(1) = lo;
n = size(pool,2);
for i = 2:n,
    temp(i) = min(pool(pool>temp(i-1)));
    if temp(i) == hi
        break
    end % if
end  % for
n = size(temp,2);

err1 = zeros(1, n+1);
err2 = zeros(1, n+1);

% Count the errors for each guess
for i=1:n,
    err1(i) = sum(set1 >= temp(i));
    err2(i) = sum(set2 < temp(i));
end  % for
err1(n+1) = sum(set1 > temp(n));
err2(n+1) = sum(set2 <= temp(n));

% Find the minimum total error
total = err1 + err2;
error = min(total);
index = find(total == error);
if size(index,2) == 1
    tidx = index;
else
    diff = abs(err1(index) - err2(index));
    mindif = min(diff);
    didx = find(diff == mindif);
    nminds = size(didx, 2);
    if nminds == 1
        tidx = index(didx);
    else
        choice = round((nminds-1) * rand) + 1;
        tidx = index(didx(choice));
    end  % if nminds
end  % if size(index,2)

if tidx > 1 & tidx <= n
    threshold = 0.5 * temp(tidx) + 0.5 * temp(tidx-1);
elseif tidx == 1
    lower = max([max(set1(set1<temp(1))) max(set2(set2<temp(1)))]);
    if size(lower,2) == 0
        threshold = temp(1) - realmin;
    else
        threshold = 0.5 * temp(1) + 0.5 * lower;
    end  % if size(lower, 2)
else
    higher = min([min(set1(set1>temp(n))) min(set2(set2>temp(n)))]);
    if size(higher,2) == 0
        threshold = temp(n) + realmin;
    else
        threshold = 0.5 * temp(n) + 0.5 * higher;
    end  % if size(higher, 2)
end  % if tidx...
```

Appendix D. Modified Code (ROC Code)

D.1 driver script file

```
% Driver script file to run the modified code and create ROC curves

load targetset;
load backgroundset;

h= 1;                          % the best h & k from the original code's output
k= 20;
[rhitp,rfap,rhitk,rfak,lhitp,lfap,lhitk,lfak]=pknn2(targetset,backgroundset,h,k);
[faave,rhitpave,rhitkave,lhitpave,lhitkave,rpu,rpl,rku,rkl,lpu,lpl,lku,lkl]=roccurve(rhitp,rfap,rhitk,rfak,
   lhitp,lfap,lhitk,lfak);
x=faave;

figure(1)
plot(Pfa,Phit,'y*')
hold on
plot(x,rhitpave,'r--',x,lhitpave,'b-',x,rpu,'r:',x,rpl,'r:',x,lpu,'b-.',x,lpl,'b-.')
axis([0 1 0 1])
xlabel('Probability of False Alarm');
ylabel('Probability of Hit');
title('Average ROC Curve (Parzen)');
legend('r--','Resubstitution','r:','R bounds','b-','Leave-one-out','b-.','L bounds','y*','Actual');
hold off

figure(2)
plot(Pfa,Phit,'y*')
hold on
plot(x,rhitkave,'r--',x,lhitkave,'b-',x,rku,'r:',x,rkl,'r:',x,lku,'b-.',x,lkl,'b-.')
axis([0 1 0 1])
xlabel('Probability of False Alarm');
ylabel('Probability of Hit');
title('Average ROC Curve (k-NN)');
legend('r--','Resubstitution','r:','R bounds','b-','Leave-one-out','b-.','L bounds','y*','Actual')
hold off

% area under leave-one-out ROC
ak=sum(lhitkave)/length(lhitkave);
ap=sum(lhitpave)/length(lhitpave);
```

D.2 pknn2.m

```
% PKNN2:  Run Parzen and kNN procedure for X1, X2 n times.
%
% Inputs:  X1, X2:  data sets (n x N1 and n x N2)
%          h, k:    values to use for h and k
%
%
% All h's must be greater than zero, and k must be between 2 and
% min(N1, N2)-1
%
% Outputs: rhitp: Probability of hit for Parzen resubstitution
%          rfap : Probability of false alarm for Parzen resubstitution
%          rhitk: Probability of hit for k-NN resubstitution
%          rfak : Probability of false alarm for k-NN resubstitution
%          lhitp: Probability of hit for Parzen leave-one-out
%          lfap : Probability of false alarm for Parzen leave-one-out
%          lhitk: Probability of hit for k-NN leave-one-out
%          lfak : Probability of false alarm for k-NN leave-one-out
%

function[rhitp,rfap,rhitk,rfak,lhitp,lfap,lhitk,lfak]= pknn2(X1, X2, h, k)

[n1, N1] = size(X1);
[n2, N2] = size(X2);
if n1 ~= n2
    fprintf(2, 'Data sets X1 and X2 must have same number of rows
(features)\n');
    return;
end  % if

% Keep this value on hand
dim = n1;
ntests = 10;

fprintf(1, 'Performing %d independent tests: \n', ntests);

rhitp=[];
rfap=[];
rhitk=[];
rfak=[];
lhitp=[];
lfap=[];
lhitk=[];
lfak=[];

counter=0;

for test = 1:ntests,

    fprintf(1,' Test #%d:\n', test);

    testsamps = test:ntests:N1;
    t1 = [1, testsamps+1];
    t2 = [testsamps-1, N1];
    others = [];
    for i = 1:size(t1,2),
        others = [others, t1(i):t2(i)];
    end  % for i

    x1 = X1(:, testsamps);
    iS1 = cov(X1(:,others)');

    testsamps = test:ntests:N2;
    t1 = [1, testsamps+1];
```

```
t2 = [testsamps-1, N2];
others = [];
for i = 1:size(t1,2),
    others = [others, t1(i):t2(i)];
end  % for i

x2 = X2(:, testsamps);
iS2 = cov(X2(:,others)');

clear testsamps others t1 t2

n1 = size(x1, 2);
n2 = size(x2, 2);
fprintf(1, '  %d Class 1 samples, %d Class 2 samples\n', n1, n2);

fprintf(1, '  Estimating and inverting covariance matrices ...\n');
detratio = -0.5 * log(det(iS2)/det(iS1));
iS1 = inv(iS1);
iS2 = inv(iS2);

fprintf(1, '  Computing distances ...\n');

[d11, d12, d21, d22] = compute_distances(x1, x2, iS1, iS2);

fprintf(1, '  Classifying (Parzen) ...\n');

for r = h,
    counter=counter+1;
    % Compute sums:
    temp = -0.5 / r^2;
    s11 = sum(exp(temp * d11));
    s12 = sum(exp(temp * d12));
    s21 = sum(exp(temp * d21));
    s22 = sum(exp(temp * d22));

    lr1 = detratio - log((n2 * s11) ./ (n1 * s21));
    lr2 = detratio - log((n2 * s12) ./ (n1 * s22));

    ll1 = detratio - log((n2 * (s11 - 1)) ./ ((n1 - 1) * s21));
    ll2 = detratio - log(((n2 - 1) * s12) ./ (n1 * (s22 - 1)));

    [Rhit,Rfa,Lhit,Lfa] = classify2(lr1, lr2, ll1, ll2);

    lRhit=length(Rhit);
    lRfa=length(Rfa);
    rhitp(test,1:lRhit)=Rhit;
    rfap(test,1:lRfa)=Rfa;

    lLhit=length(Lhit);
    lLfa=length(Lfa);
    lhitp(test,1:lLhit)=Lhit;
    lfap(test,1:lLfa)=Lfa;

end  % for r

fprintf(1,'  Classifying (k-NN) ...\n');

% sort distances
d11 = sort(d11);
d12 = sort(d12);
d21 = sort(d21);
d22 = sort(d22);

tr = detratio + log(n1/n2);
tl1 = detratio + log((n1-1)/n2);
```

```
    tl2 = detratio + log(n1/(n2-1));

    for i = k,
        lr1 = tr + 0.5 * dim * log(d11(i,:) ./ d21(i,:));
        lr2 = tr + 0.5 * dim * log(d12(i,:) ./ d22(i,:));

        ll1 = tl1 + 0.5 * dim * log(d11(i+1,:) ./ d21(i,:));
        ll2 = tl2 + 0.5 * dim * log(d12(i,:) ./ d22(i+1,:));

        [Rhit,Rfa,Lhit,Lfa] = classify2(lr1, lr2, ll1, ll2);

        lRhit=length(Rhit);
        lRfa=length(Rfa);
        rhitk(test,1:lRhit)=Rhit;
        rfak(test,1:lRfa)=Rfa;

        lLhit=length(Lhit);
        lLfa=length(Lfa);
        lhitk(test,1:lLhit)=Lhit;
        lfak(test,1:lLfa)=Lfa;

    end  % for i

end  % for test
```

D.3 classify2.m

```
% CLASSIFY2: classify resubstitution and leave-one-out
%            discriminant values.
%
% Inputs:  Lr1, Lr2: resubstitution discriminant values
%          Ll1, Ll2: leave-one-out discriminant values
%
% Outputs: Rhit,Rfa: Resubstitution probabilities
%          Lhit,Lfa: Leave-one-out probabilities
%

function [Rhit,Rfa,Lhit,Lfa] = classify2(Lr1, Lr2, Ll1, Ll2)

% First get the minimum resubstitution error
fprintf(1,' Resubstitution...');
[Rhit,Rfa] = min_error2(Lr1, Lr2);
fprintf(1,'done.\n');

% Find the leave-one-out error
fprintf(1,' Leave one out...');
[Lhit,Lfa]= min_error2(Ll1,Ll2);
fprintf(1,'done.\n');
```

D.4 min_error2.m

```
% MIN_ERROR2: Given two sets of discriminant values, finds the probabilities
%              of hit and false alarm
%
% Inputs:  SET1, SET2: sets of discriminants from classes 1 & 2
%
% Outputs: Phit: Probability of a hit
%          Pfa : Probability of false alarm
%

function [Phit, Pfa] = min_error2(set1, set2)

i=find(set1~=NaN);
temp=set1(i);
set1=temp;

j=find(set2~=NaN);
temp2=set2(j);
set2=temp2;

clear i j temp temp2

big = realmax - realmin;
search = 0;

% Eliminate infinities from the search.
infs = find(set1==inf);
set1(infs) = big * ones(size(infs));
infs = find(set1==(-inf));
set1(infs) = -big * ones(size(infs));
infs = find(set2==inf);
set2(infs) = big * ones(size(infs));
infs = find(set1==(-inf));
set2(infs) = -big * ones(size(infs));

a = min(set1);
b = max(set1);
c = min(set2);
d = max(set2);


% determines how the discriminant values of the two sets overlap
if ((b < c) | (d < a))
    Phit=1;
    Pfa=0;
    return;
end  % if


if (a <= c & c <= d & d <= b)
    lo = c;
    hi = d;
elseif ( c <= a & a <= b & b <= d )
    lo = a;
    hi = b;
elseif (a <= c & c <= b & b <= d)
    lo = c;
    hi = b;
elseif (c <= a & a <= d & d <= b)
    lo = c;
    hi = b;
else
    % What's left??? bad decision rules.
    fprintf(1,'Bad decision');
```

```
    Phit=.5;
    Pfa=.5;
    return;
end  % if

% go fish (in a limited pool)
pool = [set1(set1>=lo & set1<=hi) set2(set2>=lo & set2<=hi)];

% Get one copy of each distinct value in temp
temp(1) = lo;
n = size(pool,2);
for i = 2:n,
    temp(i) = min(pool(pool>temp(i-1)));
    if temp(i) == hi
        break
    end % if
end % for
n = size(temp,2);

err1 = zeros(1, n+1);
err2 = zeros(1, n+1);

% Count the errors for each guess
   for i=1:n,
      err1(i) = sum(set1 >= temp(i));
      err2(i) = sum(set2 < temp(i));
   end  % for
   err1(n+1) = sum(set1 > temp(n));
   err2(n+1) = sum(set2 <= temp(n));

% Find the probabilities
Pmiss=err1./length(set1);
Phit=1 - Pmiss;
Pfa=err2./length(set2);
```

D.5 roccurve.m

```
% ROCCURVE :  Creates the ROC curves from the hit and false alarm probabilities.
%
% Inputs:  rhitp,rhitk,lhitp,lhitk : hit probability matrices
%          rfap,rfak,lfap,lfak : corresponding false alarm matrices
%
% Outputs: faave : average false alarm probabilities vector
%          rhitpave,rhitkave,lhitpave,lhitkave : average hit probabilities
%          rpu,rku,lpu,lku : upper bound on 95% confidence interval
%          rpl,rkl,lpl,lkl : lower bound on 95% confidence interval
%

function[faave,rhitpave,rhitkave,lhitpave,lhitkave,rpu,rpl,rku,rkl,lpu,lpl,lku,lkl]=roccurve(rhitp,rfap,
   rhitk,rfak,lhitp,lfap,lhitk,lfak)

[r,c]=size(rhitp);

x=0:.01:1;
faave=x;

rhitp1=interpolate(rfap(1,:),rhitp(1,:),x);
rhitp2=interpolate(rfap(2,:),rhitp(2,:),x);
rhitp3=interpolate(rfap(3,:),rhitp(3,:),x);
rhitp4=interpolate(rfap(4,:),rhitp(4,:),x);
rhitp5=interpolate(rfap(5,:),rhitp(5,:),x);
rhitp6=interpolate(rfap(6,:),rhitp(6,:),x);
rhitp7=interpolate(rfap(7,:),rhitp(7,:),x);
rhitp8=interpolate(rfap(8,:),rhitp(8,:),x);
rhitp9=interpolate(rfap(9,:),rhitp(9,:),x);
rhitp10=interpolate(rfap(10,:),rhitp(10,:),x);

rhitk1=interpolate(rfak(1,:),rhitk(1,:),x);
rhitk2=interpolate(rfak(2,:),rhitk(2,:),x);
rhitk3=interpolate(rfak(3,:),rhitk(3,:),x);
rhitk4=interpolate(rfak(4,:),rhitk(4,:),x);
rhitk5=interpolate(rfak(5,:),rhitk(5,:),x);
rhitk6=interpolate(rfak(6,:),rhitk(6,:),x);
rhitk7=interpolate(rfak(7,:),rhitk(7,:),x);
rhitk8=interpolate(rfak(8,:),rhitk(8,:),x);
rhitk9=interpolate(rfak(9,:),rhitk(9,:),x);
rhitk10=interpolate(rfak(10,:),rhitk(10,:),x);

lhitp1=interpolate(lfap(1,:),lhitp(1,:),x);
lhitp2=interpolate(lfap(2,:),lhitp(2,:),x);
lhitp3=interpolate(lfap(3,:),lhitp(3,:),x);
lhitp4=interpolate(lfap(4,:),lhitp(4,:),x);
lhitp5=interpolate(lfap(5,:),lhitp(5,:),x);
lhitp6=interpolate(lfap(6,:),lhitp(6,:),x);
lhitp7=interpolate(lfap(7,:),lhitp(7,:),x);
lhitp8=interpolate(lfap(8,:),lhitp(8,:),x);
lhitp9=interpolate(lfap(9,:),lhitp(9,:),x);
lhitp10=interpolate(lfap(10,:),lhitp(10,:),x);

lhitk1=interpolate(lfak(1,:),lhitk(1,:),x);
lhitk2=interpolate(lfak(2,:),lhitk(2,:),x);
lhitk3=interpolate(lfak(3,:),lhitk(3,:),x);
lhitk4=interpolate(lfak(4,:),lhitk(4,:),x);
lhitk5=interpolate(lfak(5,:),lhitk(5,:),x);
lhitk6=interpolate(lfak(6,:),lhitk(6,:),x);
lhitk7=interpolate(lfak(7,:),lhitk(7,:),x);
lhitk8=interpolate(lfak(8,:),lhitk(8,:),x);
lhitk9=interpolate(lfak(9,:),lhitk(9,:),x);
lhitk10=interpolate(lfak(10,:),lhitk(10,:),x);
```

```
% averages
rhitpave=(rhitp1+rhitp2+rhitp3+rhitp4+rhitp5+rhitp6+rhitp7+rhitp8+rhitp9+rhitp10)/10;
rhitkave=(rhitk1+rhitk2+rhitk3+rhitk4+rhitk5+rhitk6+rhitk7+rhitk8+rhitk9+rhitk10)/10;
lhitpave=(lhitp1+lhitp2+lhitp3+lhitp4+lhitp5+lhitp6+lhitp7+lhitp8+lhitp9+lhitp10)/10;
lhitkave=(lhitk1+lhitk2+lhitk3+lhitk4+lhitk5+lhitk6+lhitk7+lhitk8+lhitk9+lhitk10)/10;

%standard deviations
rp=[rhitp1;rhitp2;rhitp3;rhitp4;rhitp5;rhitp6;rhitp7;rhitp8;rhitp9;rhitp10];
rk=[rhitk1;rhitk2;rhitk3;rhitk4;rhitk5;rhitk6;rhitk7;rhitk8;rhitk9;rhitk10];
lp=[lhitp1;lhitp2;lhitp3;lhitp4;lhitp5;lhitp6;lhitp7;lhitp8;lhitp9;lhitp10];
lk=[lhitk1;lhitk2;lhitk3;lhitk4;lhitk5;lhitk6;lhitk7;lhitk8;lhitk9;lhitk10];

srp=std(rp);
srk=std(rk);
slp=std(lp);
slk=std(lk);


% confidence intervals
rpu=rhitpave+2.262*(srp./sqrt(10));
rpl=rhitpave-2.262*(srp./sqrt(10));

rku=rhitkave+2.262*(srk./sqrt(10));
rkl=rhitkave-2.262*(srk./sqrt(10));

lpu=lhitpave+2.262*(slp./sqrt(10));
lpl=lhitpave-2.262*(slp./sqrt(10));

lku=lhitkave+2.262*(slk./sqrt(10));
lkl=lhitkave-2.262*(slk./sqrt(10));
```

D.6 interpolate.m

```
% INTERPOLATE:  Interpolates points on the given curves
%
% Inputs:   fa  : false alarm probabilities
%           hit : corresponding hit probabilities
%
% Output:   newhit2 : interpolated hit vector, same length as newvector
%

function[newhit2]=interpolate(fa,hit,newvector)

fa=(round(1000*fa))/1000;
s=max(fa);

newfa=[0];
newhit=[0];

y=find(fa==0);                                  % gets hit value for fa=0
if isempty(y)==0
   z=find(hit(y));
   if isempty(z)==0
      yy=(sum(hit(z)))/length(z);
      newhit=[yy];
   end
end

for i=.001:.001:s                               % creates monotonic points
   y=find(fa==i);                               % to put into interp1
   if isempty(y)==0
      newfa=[newfa i];
      yy=(sum(hit(y)))/length(y);
      newhit=[newhit yy];
   end
end

if s~=1
   newfa=[newfa 1];                             % final value for fa=1
   newhit=[newhit 1];
end

newhit2=interp1(newfa,newhit,newvector);        % does interpolation
```

Bibliography

1. Casasent, David P. "Sequential and Fused Optical Filters for Clutter Reduction and Detection." *SPIE-The International Society for Optical Engineering1959*. April 1993.
2. Deming, W. Edwards. *Sample Design in Business Research*. New York: Wiley, 1960.
3. Donohoe, Gregory W. *Image Processing Short Course with Khoros*, 1992.
4. Duda, Richard O. and Peter E. Hart. *Pattern Classification and Scene Analysis*. New York: John Wiley and Sons, 1973.
5. Egan, James P. *Signal Detection Theory and ROC Analysis*. New York: Academic Press, 1975.
6. Fix, E. and J.L. Hodges. *Discriminatory Analysis, Nonparametric Discrimination, Consistency Properties*. Volume 4, Project 21-49-004, Randolph Field, TX: School of Aviation Medicine, 1951.
7. Fukunaga, Keinosuke and Donald M. Hummels. "Bayes Error Estimation Using Parzan and k-NN Procedures," *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9* (1987).
8. Gonzalez, Rafael C. and Richard E. Woods. *Digital Image Processing*. Reading, Mass: Addison-Wesley Publishing Co., 1992.
9. Hanson, M.H., W.N. Hurwitz and W.G. Madow. *Sample Survey Methods and Theory*. New York: Wiley, 1953.
10. Lachenbruch, Peter A. "An Almost Unbiased Method of Obtaining Confidence Intervals for the Probability of Misclassification in Discriminant Analysis," *Biometrics*, *23* (1967).
11. Martin, Curtis E. *Non-Parametric Bayes Error Estimation for UHRR Target Identification*. Masters Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
12. Martin, Curtis E., Steven K. Rogers and Dennis W. Ruck. "Neural Network Bayes Error Estimation." *IEEE ICNN1*. 305–308. 1994.
13. Metz, Charles E. "ROC Methodology in Radiologic Imaging," *Investigative Radiology*, *21*(9):720–733 (September 1986).
14. Papoulis, Athanasios. *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, Inc., 1991.
15. Parzen, Emanuel. "On Estimation of a Probability Density Function and Mode," *Ann. Math. Stat.*, *33* (1962).
16. Sklar, Bernard. *Digital Communications Fundamentals and Applications*. New Jersey: PTR Prentice Hall, 1988.

17. Snedecor, George W. and William G. Cochran. *Statistical Methods.* Ames, Iowa: The Iowa State University Press, 1980.

18. Thomas, John B. *Introduction to Probability.* New York: Dowden and Culver Inc., 1986.

19. Wald, A. *Statistical Decision Functions.* New York: Wiley, 1950.

Vita

Lieutenant Georgia K. Harrup was born Georgia K. Ploegman on 19 June 1971 in Orlando, Florida. She attended Michigan Technological University where she graduated in 1993 with a Bachelor of Science Degree in Electrical Engineering. Upon graduation, she accepted a commission in the United States Air Force. In June 1993, her first active duty assignment was to the Air Force Institute of Technology, where she studied for a Master of Science Degree in Electrical Engineering.

Permanent address: 21 W Parkwood Dr. Apt C
Fairborn, OH 45324

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Master's Thesis

4. TITLE AND SUBTITLE
ROC Analysis of IR Segmentation Techniques

5. FUNDING NUMBERS

6. AUTHOR(S)
Georgia K. Harrup

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/94D-15

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Capt. Thomas Burns
USAF Wright Laboratory
WL/AARA
WPAFB, OH 45433-7001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Receiver Operating Characteristic (ROC) curves are used to compare the effectiveness of IR image processing techniques. Two non-parametric error estimation techniques (k-Nearest Neighbor and Parzen Window) are used to create estimates of the probability density functions for the data. These pdfs are used in the creation of the ROC curves for both resubstitution and leave-one-out estimates. These estimates generate the upper and lower bounds, respectively, on the ROC curves. The ROC curve analysis is performed on the outputs of various image processing techniques and the resulting ROC curves are used to compare the techniques. Of the image processing techniques used in this thesis, the close minus open (CMO) morphological filter operation produced the best results.

14. SUBJECT TERMS
Receiver Operating Characteristic (ROC), Morphology, Spatial Feature Extraction, Non-Parametric Density Estimation

15. NUMBER OF PAGES
111

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL